# Towards a "NoOps" Model for WLCG

Rob Gardner
University of Chicago

CHEP 2019
November 5, 2019

## After 20 years of grid deployments, can we improve our deployment model?

Worldwide LHC Computing Grid

## But our current model works just fine?

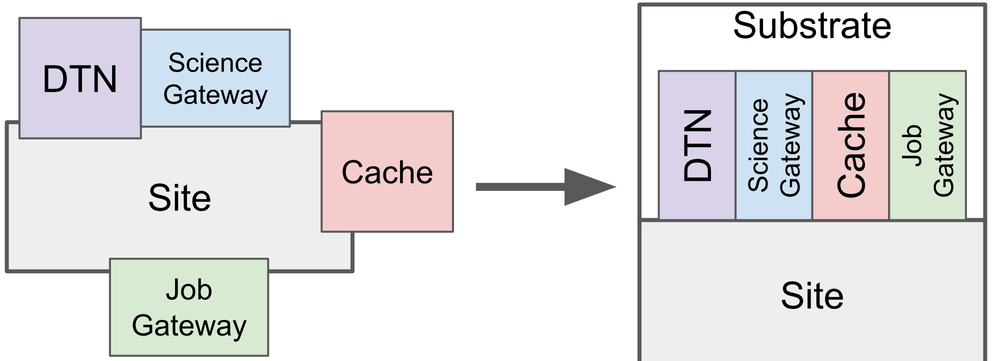
- We have >150 sites integrated into a production fabric capable of delivering billions of CPU-hours and transferring hundreds of PB/year
- However we know its costly to operate, difficult to innovate, thus slow to change
- Meanwhile the world is moving on - new ecosystems, software delivery pipelines, frameworks, ops models
- And our software is being "upgraded" to meet the enormous challenge of the HL-LHC era

## Lets talk about re-federating the edge

- In the WLCG edge of **today** we have
  - **Network diagnostics** (e.g. PerfSONAR)
  - **Data transfer (storage)** endpoints
  - **Compute elements** to route jobs
  - **Software & conditions data caches** (CVMFS, Frontier-squid)
  - **Data caches** (e.g. Xrootd Cache)
- In future?
  - New data delivery services
  - Facility API's (IaaS)
  - Platform APIs (PaaS)
  - ?

# Standardizing an Edge Service Substrate

- Currently, each piece of infrastructure added to a site tends to require
  - a person located at the site to advocate for setting it up and to manage it
  - a 'hand-built' custom installation, local configuration management
- By adding a **consistent edge substrate** that is common to sites and modular service components which use it, labor can be reduced
- Security challenges change, though, because instead of considering one service to permit (at a time), the site must consider the whole substrate

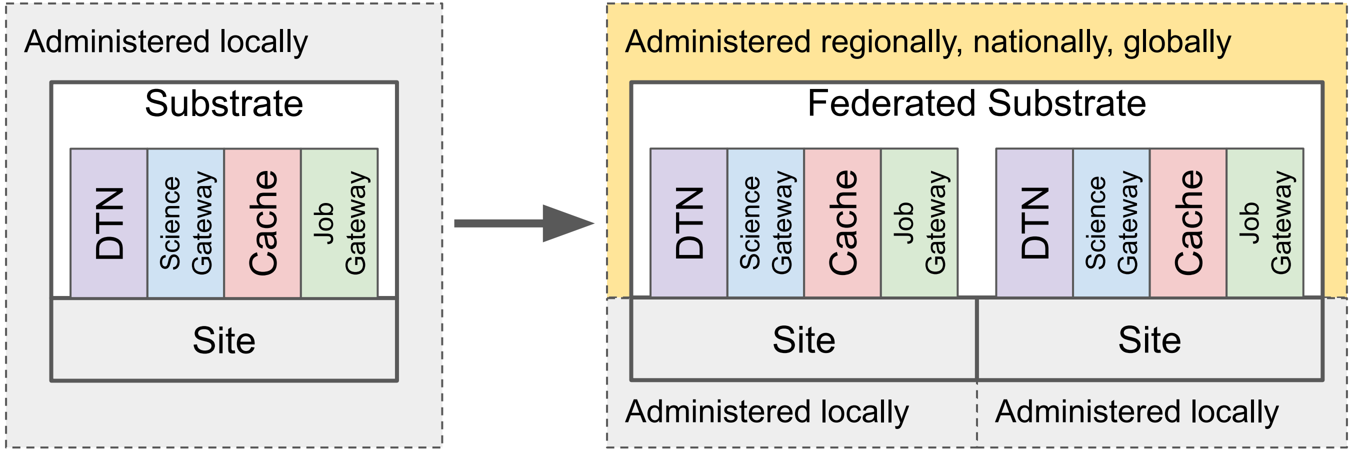


## Possible Approaches

- An edge substrate can do more than just standardize single sites: It can be distributed, giving a single interface to address many sites
- A distributed substrate can be federated in different ways:
  - Hardware deployed at each site may be managed centrally
    - This is, as we understand it, broadly the approach taken by the Pacific Research Platform (PRP)
  - Hardware may be controlled by local site admins, who then grant fine-grained permissions to external organizations
    - This is the approach we would like to expand on in this talk
- Different methods may be better suited to different collections of sites and different end uses
  - A simpler, centralized platform probably works best for some
  - Some sites (national labs, for example), have indicated that they would require greater local control

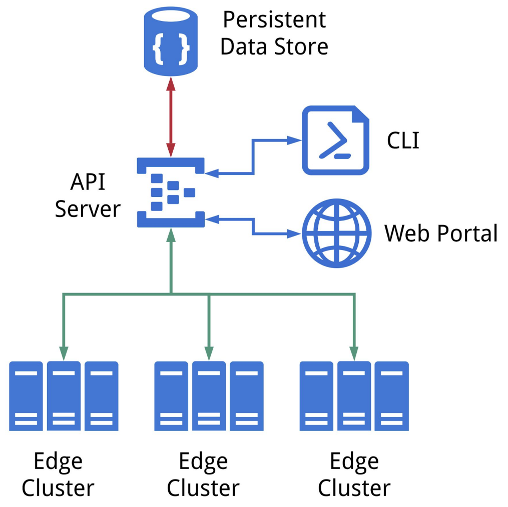
## Federated platforms present new challenges

- Services have traditionally only been the responsibility of the local admin and security teams
- Building **multi-site platforms** for orchestrating services means that:
  - Sites need to define or review policies for external administration of services
  - Platforms need to establish their policies for interacting with sites and define how they will use resources



## The SLATE Platform for Edge Services

- SLATE (Services Layer at the Edge) provides a substrate for this type of infrastructure
- Docker, Kubernetes, and Helm are used to package and deploy service applications
- A central server component is used to mediate user requests being sent to participating edge Kubernetes clusters
- State is stored persistently in DynamoDB, with sensitive data encrypted while 'at rest'
- Command line and web interfaces are provided



slateci.io

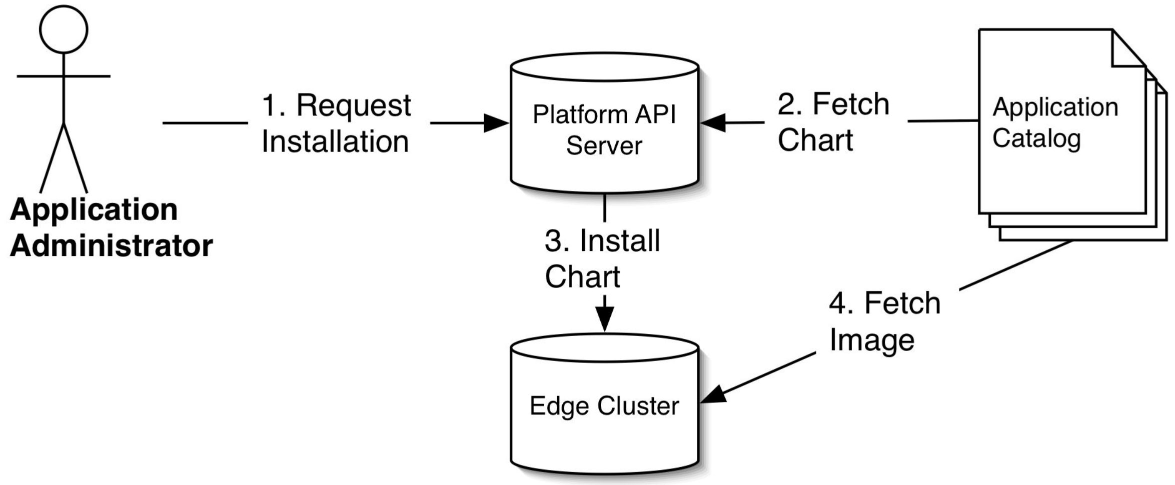
## Evolving Trust and Privilege in WLCG

- Infrastructure services are qualitatively different from the batch jobs that many sites already accept from outside users—they must run persistently, and must often accept network connections from outside
- To admit such services, site administrators need strong guarantees that:
  - Only suitable persons will be able to deploy services
  - Only appropriate software for providing a relevant service will be run
  - Services will use appropriately secure software and configuration
  - External users running services will not interfere with existing uses of resources
- Users running services also want separation between their applications and others'

## Application (containerized service) Packaging

- SLATE makes use of Helm to package applications for Kubernetes
  - Helm is commonly used in the broader Kubernetes community
  - Helm enables templating Kubernetes YAML manifests for more convenient reuse
- Only limited configuration settings for each application are exposed by its Helm chart
  - Hides complexity users don't want to see
  - Can be used to enforce required aspects of configuration
  - Provides a consistent interface which all participants in the federation can inspect and agree on
- SLATE maintains its own catalog of charts, and allows only those applications to be installed
- In future, a WLCG federation may curate & maintain such apps

# Application Install Process



- The SLATE API server mediates requests to install applications
  - Fetches applications only from the curated catalog
  - Enforces rules set by the administrators of the target cluster

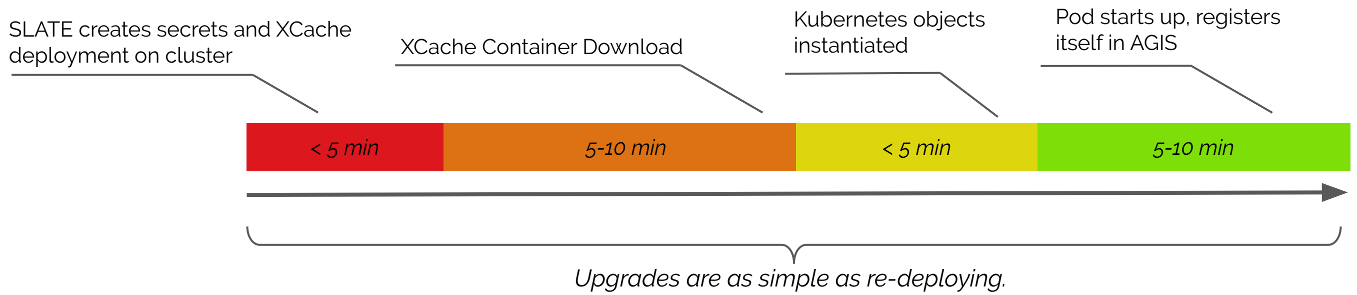
## Deployment experience in ATLAS

- Goal: build an XCache-based caching network as part of the DOMA activity
- SLATE-registered Kubernetes clusters operational at a number of ATLAS sites MWT2, AGLT2, LRZ
  - ESnet Sunnyvale for a short duration test
- XCache application curated in the SLATE catalog

## How it worked

- Register a cluster with SLATE and allow the `atlas-xcache` group
- Apply a few special extra steps for XCache:
  - Node labeled in Kubernetes (`xcache-capable=true`)
  - One or more storage volumes mounted (e.g. `/xcache`) & communicated to deployer
  - Endpoint protocol registered in AGIS (ATLAS info service)
- Test suite containerized
  - Launch a very realistic stress test from Google Compute Engine in minutes

**<u>XCache Deployment & Upgrade Cycle</u>:**


SLATE creates secrets and XCache deployment on cluster
XCache Container Download
Kubernetes objects instantiated
Pod starts up, registers itself in AGIS
< 5 min
5-10 min
< 5 min
5-10 min
Upgrades are as simple as re-deploying.

**XCache fully deployed in less than 20 minutes.**

## XCache update process

- Even simpler
- Completely transparent to site admin.

```
$ slate instance list
$ slate instance delete <instance name>
$ slate app install --group atlas-xcache --cluster uchicago-prod --conf MWT2.yaml xcache
```

Additional benefits:

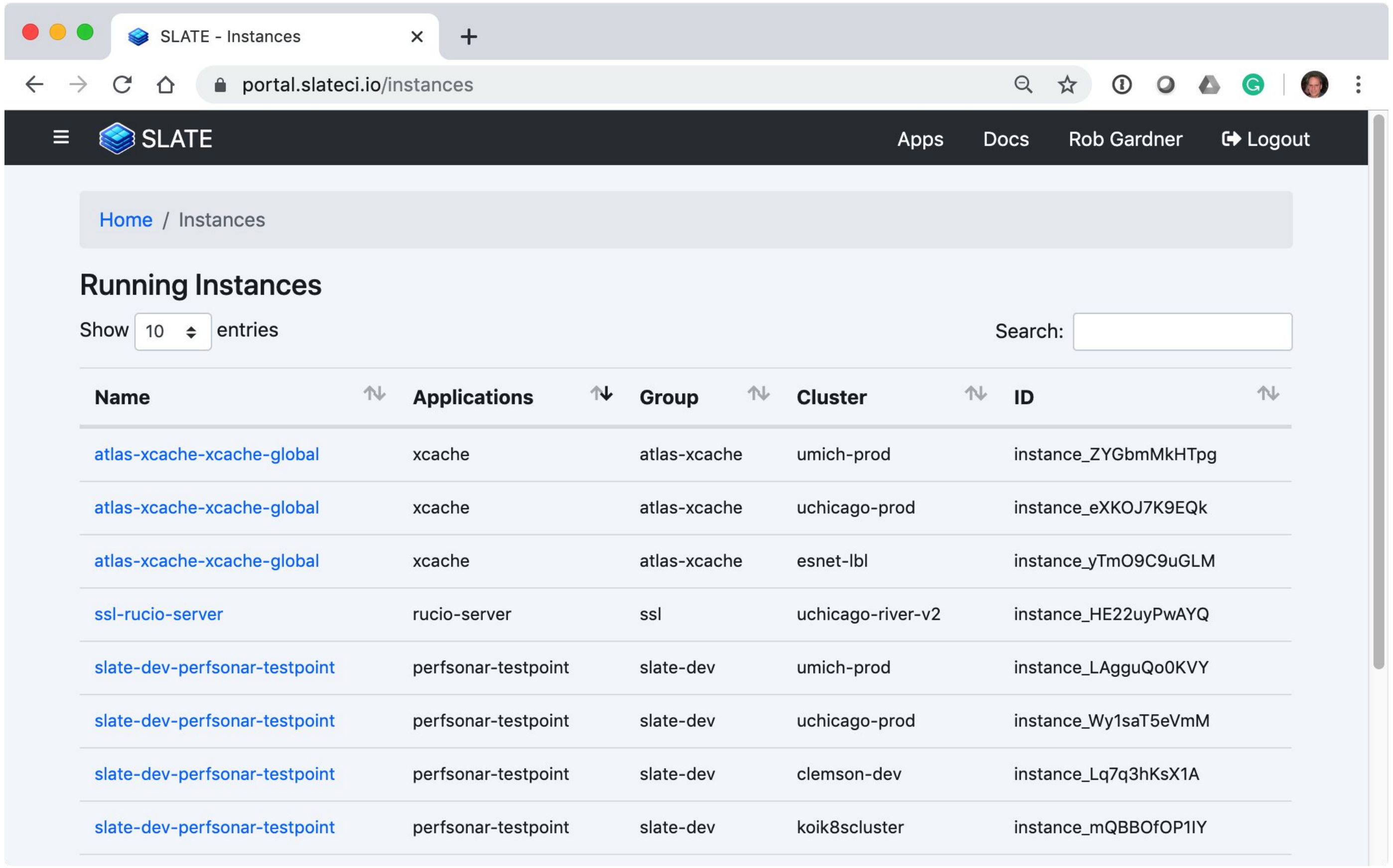
- Automatic core dump collection
- Containerized environment makes it easier to debug

## We are using SLATE to manage squids...

- OSG announced a vulnerability in Frontier Squid on July 26
- SLATE instances were all updated within the hour with this script:

```
for i in $(slate instance list | grep squid | awk '{print $4}'); do
   slate instance restart $i
done
```

# ...and other containerized services



Home / Instances

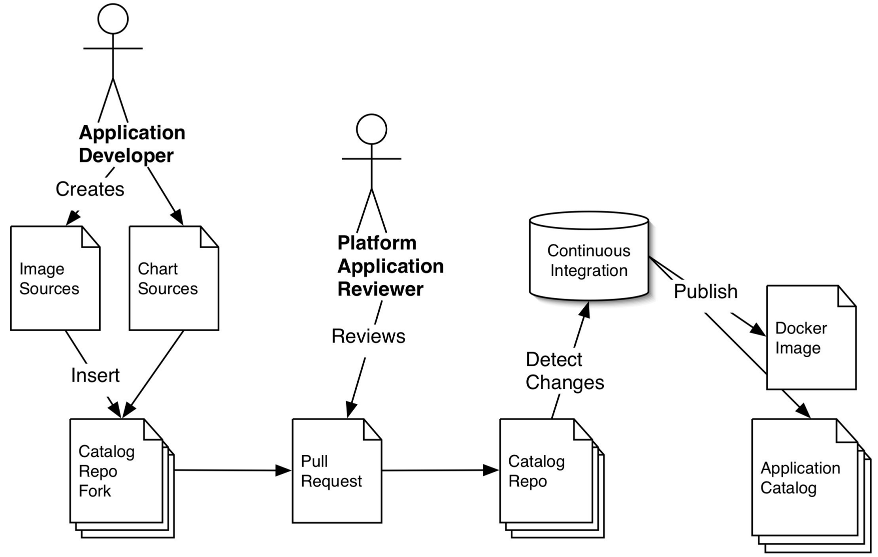
Running Instances

Show 10 entries

Search:

| Name | Applications | Group | Cluster | ID |
|---|---|---|---|---|
| atlas-xcache-xcache-global | xcache | atlas-xcache | umich-prod | instance_ZYGbmMkHTpg |
| atlas-xcache-xcache-global | xcache | atlas-xcache | uchicago-prod | instance_eXKOJ7K9EQk |
| atlas-xcache-xcache-global | xcache | atlas-xcache | esnet-lbl | instance_yTmO9C9uGLM |
| ssl-rucio-server | rucio-server | ssl | uchicago-river-v2 | instance_HE22uyPwAYQ |
| slate-dev-perfsonar-testpoint | perfsonar-testpoint | slate-dev | umich-prod | instance_LAgguQo0KVY |
| slate-dev-perfsonar-testpoint | perfsonar-testpoint | slate-dev | uchicago-prod | instance_Wy1saT5eVmM |
| slate-dev-perfsonar-testpoint | perfsonar-testpoint | slate-dev | clemson-dev | instance_Lq7q3hKsX1A |
| slate-dev-perfsonar-testpoint | perfsonar-testpoint | slate-dev | koik8scluster | instance_mQBBOfOP1IY |

# Application Curation



- Much of the value of the centralized application catalog derives from the overesight applied to the applications added to it
- Some amount of human attention is required, but maximizing automation is highly desirable

## Curation Considerations

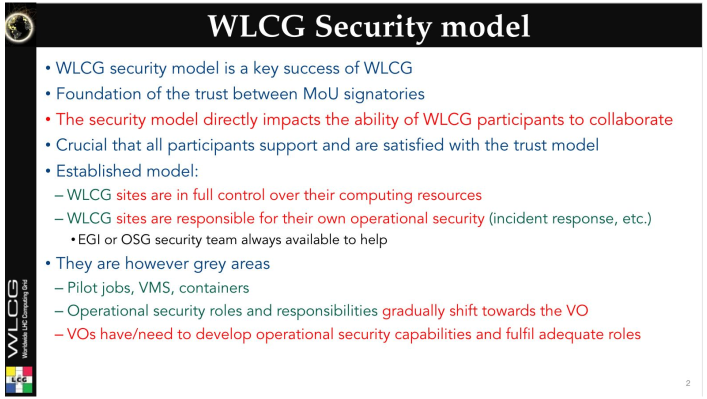
- Both Helm charts and the container images they reference must be taken into consideration
- The review process must be deep enough to be able to prevent problems, but not so slow or restrictive that potential users of the platform cannot get appropriate applications into production
  - Reviewer effort is also a limited resource!
- Some trusted sources are needed as a basis
  - The community already trusts major OS distributions (CentOS, Ubuntu, etc.)
  - Some applications are provided by major groups within the wider community which already have their own processes for trustworthy releases (Apache httpd, NGINX)

## Broader Policy Concerns

- The SLATE Team has been working on an engagement with TrustedCI, with one major goal being to design security policies and procedures
- Incident Response and Disaster Recovery have been identified as particularly critical areas
  - Incident Response, in particular can involve multiple sites, and a need to share information in a timely manner
- We think that getting these policy areas structured correctly is key building a useful platform
- Eventually, we hope to have policies which can themselves be considered sufficiently standard for broad adoption by the community
  - This means that we need to form a clear picture of what sites' concerns are
  - The WLCG (CERN Large Hadron Collider) has set up a working group to investigate these ideas as well

## WLCG SLATE Security Working Group

- Deployment of XCache with SLATE in ESnet elevated security concerns due to vulnerabilities found in software images
- In response, formed a working group charter at Fermilab pre-GDB in September (R. Wartel and myself chair)
- Follow-up at WISE workshop in San Diego, NSF Cybersecurity Summit for Large Facilities



### WLCG Security model

- WLCG security model is a key success of WLCG
- Foundation of the trust between MoU signatories
- The security model directly impacts the ability of WLCG participants to collaborate
- Crucial that all participants support and are satisfied with the trust model
- Established model:
  - WLCG sites are in full control over their computing resources
  - WLCG sites are responsible for their own operational security (incident response, etc.)
    - EGI or OSG security team always available to help
- They are however grey areas
  - Pilot jobs, VMS, containers
  - Operational security roles and responsibilities gradually shift towards the VO
  - VOs have/need to develop operational security capabilities and fulfil adequate roles

2

### In Progress and Planned Policies

| In Progress | Planned |
|---|---|
| Acceptable Use Policy<br>Asset Management Policy (part of MISPP)<br>Incident Response Procedures | Disaster Recovery Policy<br>Access Control Policy<br>Edge Admin/Federation Policy<br>Information Classification Policy (MISPP)<br>Personnel Exit Checklist<br>Mobile Compute Policy<br>Network Security Policy<br>Password Policy<br>Physical/Environmental Security Policy<br>Privacy Policy<br>Remote Access Policy (MISPP) |

## Conclusions

- Federated operations is possible today and is being advanced by a number of teams within trusted domains
- A curated application catalog important component for secure deployment
- Besides reducing operation effort at sites, the NoOps model can increase security, improve monitoring, provide faster updates, allow central opertation by experts
- Changing roles/functions at sites -- grow a containerization community
- Working on policy aspects, thus feedback from the community is essential

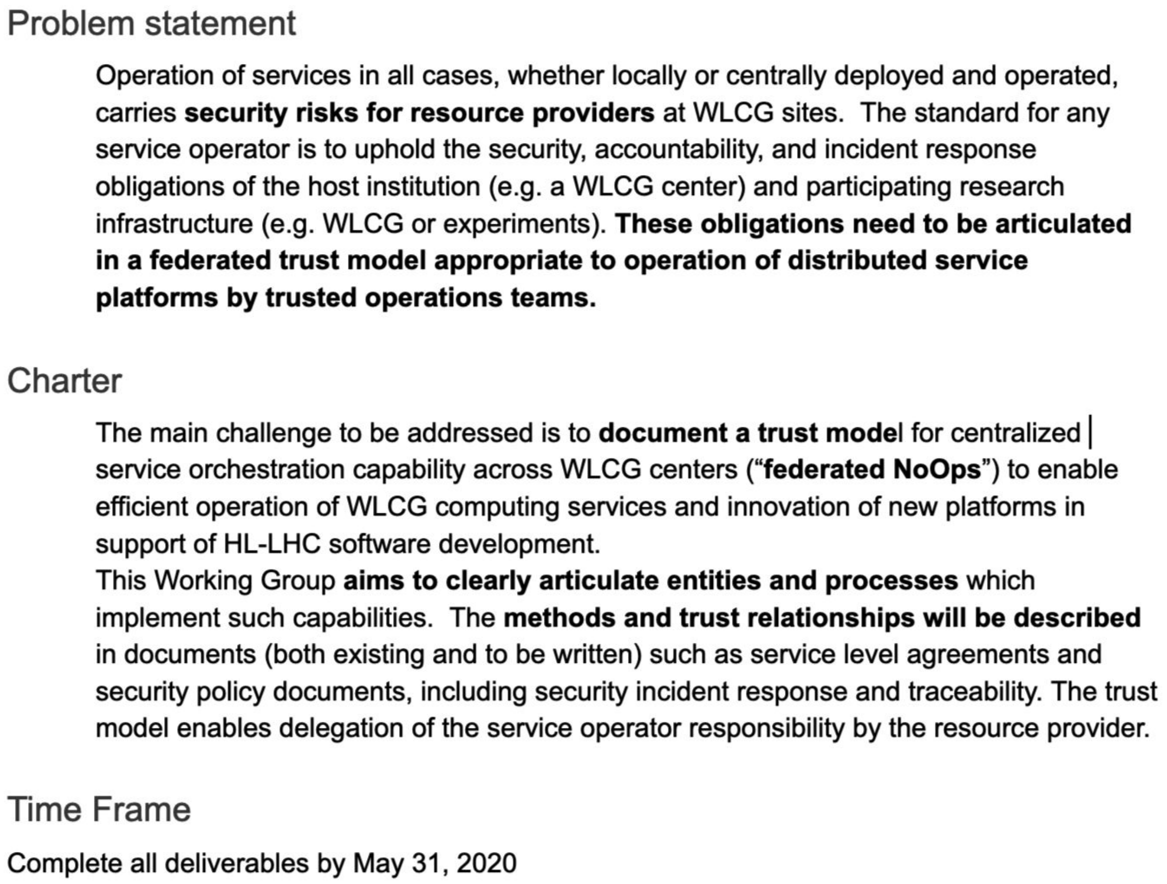
# Thanks

extra slides follow

This work is supported by the National Science Foundation Office of Advanced Cyberinfrastructure (OAC), grant number 1724821

## WLCG SLATE Security Working Group

link



### Problem statement

Operation of services in all cases, whether locally or centrally deployed and operated, carries **security risks for resource providers** at WLCG sites. The standard for any service operator is to uphold the security, accountability, and incident response obligations of the host institution (e.g. a WLCG center) and participating research infrastructure (e.g. WLCG or experiments). **These obligations need to be articulated in a federated trust model appropriate to operation of distributed service platforms by trusted operations teams.**

### Charter

The main challenge to be addressed is to **document a trust mode**l for centralized service orchestration capability across WLCG centers ("**federated NoOps**") to enable efficient operation of WLCG computing services and innovation of new platforms in support of HL-LHC software development.

This Working Group **aims to clearly articulate entities and processes** which implement such capabilities. The **methods and trust relationships will be described** in documents (both existing and to be written) such as service level agreements and security policy documents, including security incident response and traceability. The trust model enables delegation of the service operator responsibility by the resource provider.

### Time Frame

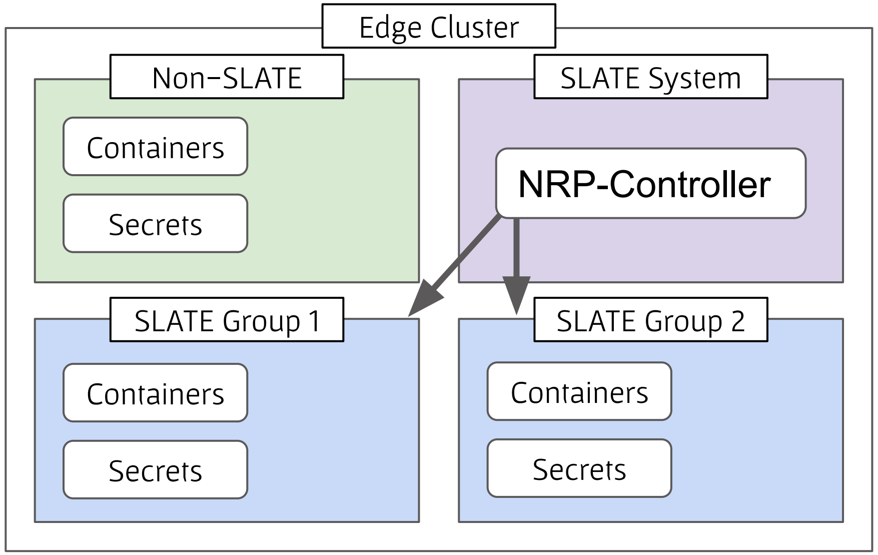
Complete all deliverables by May 31, 2020

## An Aside On Containers

- Linux Containers are a useful technology for implementing this type of platform, but are not fundamental to the general strategy of building a unified abstraction across multiple sites' resources
- It would be conceptually equivalent to build a substrate using Virtual Machines (leveraging OpenStack, for example), or using configuration management tools (such as Puppet or Ansible)
- Security concerns can be divided into two categories:
    - Those which are generally applicable to any distributed service platform, such as how users are granted access to sites' resources
    - Those which are specific to using containers to implement such a platform, such as the provenance of the container images and details of the container runtime

# Approach to Multi-tenancy

- SLATE uses Kubernetes' namespaces, secrets and implementation of Role-Based Access Control (RBAC)
- The SLATE API server is granted access only to its own subset of namespaces
- SLATE places applications belonging to different user groups into separate namespaces
- Kubernetes forbids containers in one namespace from reading secrets in other namespaces

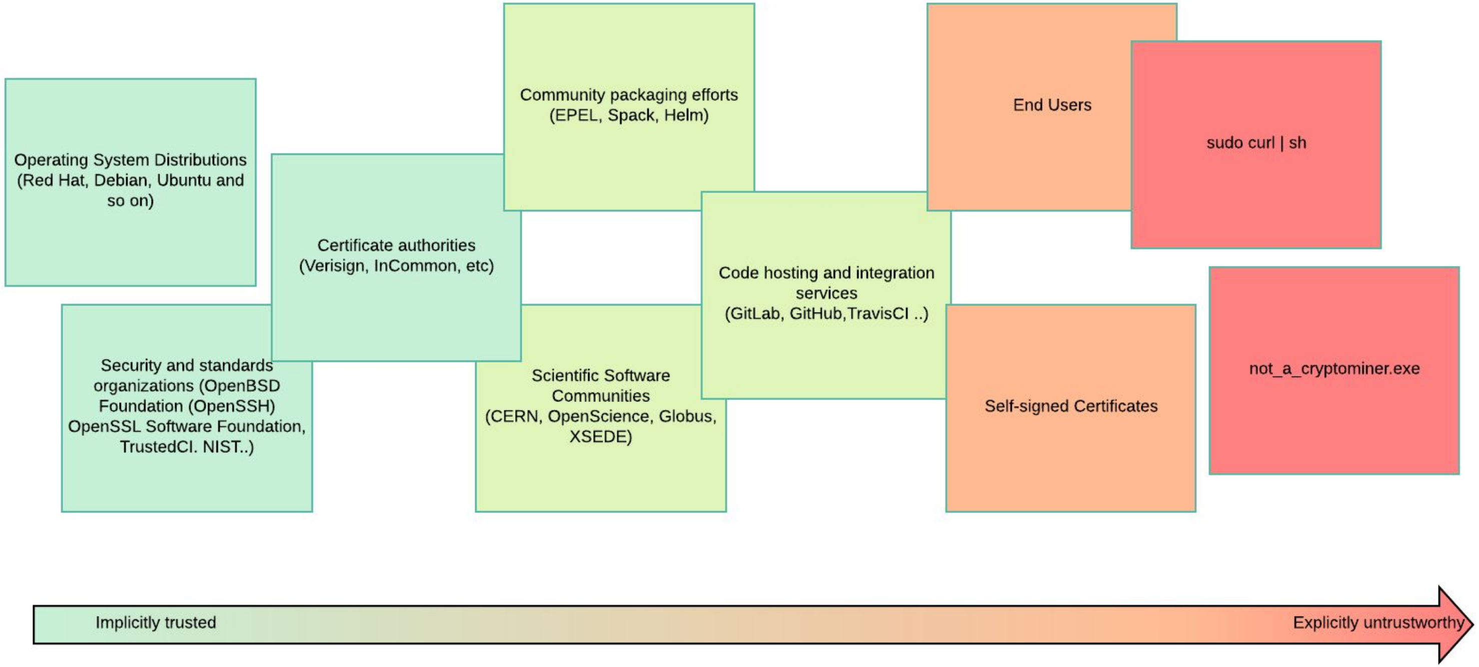


https://gitlab.com/ucsd-prp/nrp-controller

## Internal Permissions Model

- SLATE organizes users into groups, and permissions apply per-group
- Every participating cluster is administered by a group
  - When a cluster first joins the federation, only its administering group has access
- The administrators of a cluster can:
  - Grant access to other groups to deploy applications on their cluster
  - Set up per-group whitelists of which applications guest groups are authorized to deploy
- The site administrator always retains the capability to directly work with the underlying Kubernetes layer to perform actions beyond what SLATE directly supports
  - This means that local admins have no restriction on inspecting, editing, or removing components if needed

## Variation in Trust Levels


Operating System Distributions (Red Hat, Debian, Ubuntu and so on)
Security and standards organizations (OpenBSD Foundation (OpenSSH) OpenSSL Software Foundation, TrustedCI. NIST..)
Certificate authorities (Verisign, InCommon, etc)
Community packaging efforts (EPEL, Spack, Helm)
Scientific Software Communities (CERN, OpenScience, Globus, XSEDE)
Code hosting and integration services (GitLab, GitHub,TravisCI ..)
End Users
Self-signed Certificates
sudo curl | sh
not_a_cryptominer.exe
Implicitly trusted
Explicitly untrustworthy

## Special Challenges of Container Images

- Container images are a snapshot of a system state, so they do not tend to be aware of security patches since their creation
    - This implies that periodic rebuilding of images is necessary, and possibly that containers should be periodically restarted
- Typical distribution mechanisms (Docker) allow the data referred to by a particular image 'tag' to be replaced—an image which was previously reviewed may be replaced by one with different contents
    - This is why we prefer to have SLATE manage image sources, build and publish the images to a repository itself
- Automated image scanning tools can help with review, but are not a complete answer
    - Only images containing package manager data can be scanned
    - Scans may find large numbers of low-importance vulnerabilities for which no patched packages are available from the base distribution

# Risks of containers as defined by NIST

- Broadly, NIST has identified* 5 areas of risk with application containers:
    - Image Risks
        - Configuration defects, malware, embedded secrets, untrusted software
    - Registry Risks
        - Insecure connections, stale images, insufficient authentication and authorization
    - Orchestrator Risks
        - Unbounded administrative access, unauthorized access, mixed sensitivity of workloads and poor separation between workloads
    - Container Risks
        - Vulnerabilities in the Container runtime, insecure runtime configurations, unbounded network access
    - Host OS Risks
        - Large attack surface, shared kernel, host filesystem tampering, host component vulnerabilties

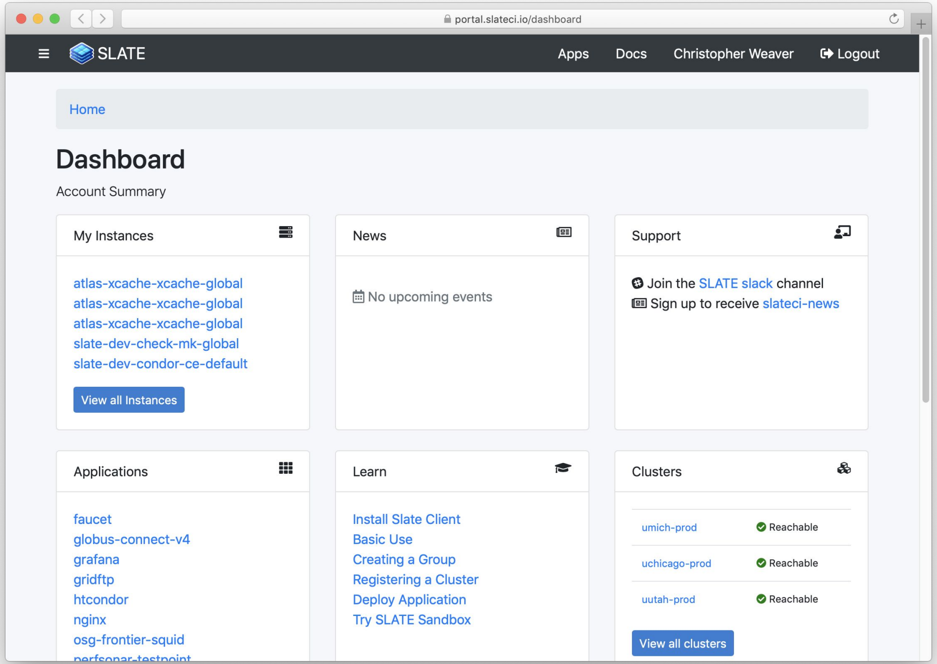
* https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf

## Risks and Mitigations

- Image Risks
  - SLATE uses a curated application catalog with specific requirements for containers that may be in the catalog.
- Registry Risks
  - The SLATE team is currently considering running a registry independent of the common ones, e.g. DockerHub to have more control over images delivered via the platform
- Orchestrator Risks
  - SLATE allows the operator of the cluster to limit which applications may be launched
  - Additionally, the Kubernetes API need only be open publicly to a few specific IPs for SLATE access
- Container (runtime) Risks / Host OS risks
  - SLATE will offer best-practices for runtime and host configuration, but largely this is left up to the Cluster administrator

* https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf

# SLATE Web Interface



Home

Dashboard

Account Summary

My Instances

atlas-xcache-xcache-global
atlas-xcache-xcache-global
atlas-xcache-xcache-global
slate-dev-check-mk-global
slate-dev-condor-ce-default

View all Instances

News

No upcoming events

Support

Join the SLATE slack channel
Sign up to receive slateci-news

Applications

faucet
globus-connect-v4
grafana
gridftp
htcondor
nginx
osg-frontier-squid
perfsonar-testpoint

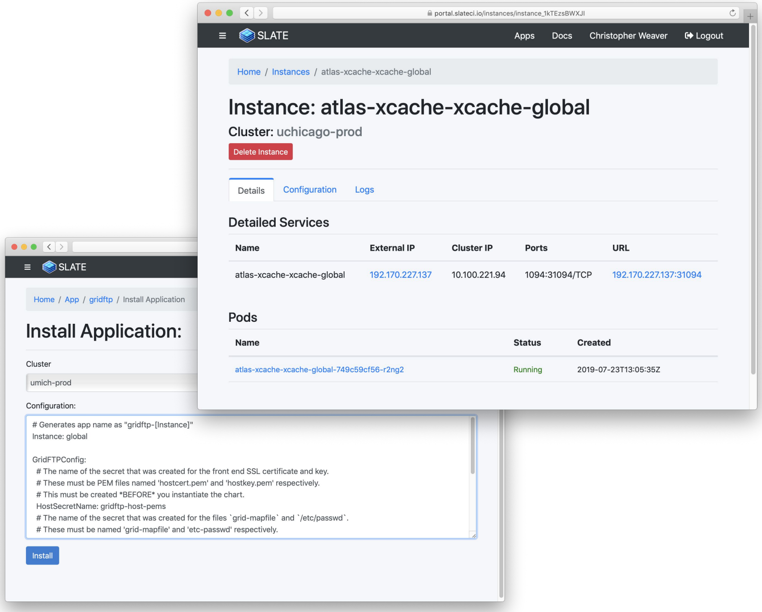
Learn

Install Slate Client
Basic Use
Creating a Group
Registering a Cluster
Deploy Application
Try SLATE Sandbox

Clusters

| | |
|---|---|
| umich-prod | Reachable |
| uchicago-prod | Reachable |
| uutah-prod | Reachable |

View all clusters



Home / Instances / atlas-xcache-xcache-global

Instance: atlas-xcache-xcache-global

Cluster: uchicago-prod

Detailed Services

| Name | External IP | Cluster IP | Ports | URL |
|---|---|---|---|---|
| atlas-xcache-xcache-global | 192.170.227.137 | 10.100.221.94 | 1094:31094/TCP | 192.170.227.137:31094 |

Pods

| Name | Status | Created |
|---|---|---|
| atlas-xcache-xcache-global-749c59cf56-r2ng2 | Running | 2019-07-23T13:05:35Z |

Home / App / gridftp / Install Application

Install Application:

Cluster

umich-prod

Configuration:

```
# Generates app name as "gridftp-[Instance]"
Instance: global

GridFTPConfig:
  # The name of the secret that was created for the front end SSL certificate and key.
  # These must be PEM files named 'hostcert.pem' and 'hostkey.pem' respectively.
  # This must be created *BEFORE* you instantiate the chart.
  HostSecretName: gridftp-host-pems
  # The name of the secret that was created for the files `grid-mapfile` and `/etc/passwd`.
  # These must be named 'grid-mapfile' and 'etc-passwd' respectively.
```

- Almost all SLATE functions are available via the portal

# Application Configuration Example

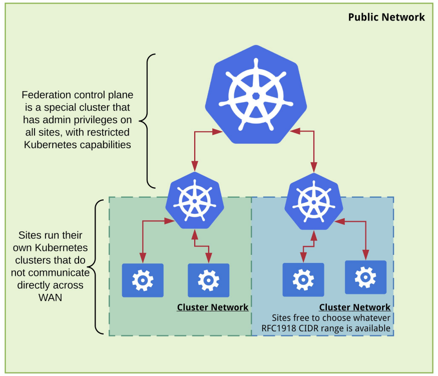
```
# Instance to label use case of Frontier Squid deployment
# Generates app name as "osg-frontier-squid-[Instance]"
# Enables unique instances of Frontier Squid in one namespace
Instance: global

SquidConf:
  # The amount of memory (in MB) that Frontier Squid may use on the machine.
  # Per Frontier Squid, do not consume more than 1/8 of system memory with Frontier Squid
  CacheMem: 128
  # The amount of disk space (in MB) that Frontier Squid may use on the machine.
  # The default is 10000 MB (10 GB), but more is advisable if the system supports it.
  # Current limit is 999999 MB, a limit inherent to helm's number conversion system.
  CacheSize: 10000
  # The range of incoming IP addresses that will be allowed to use the proxy.
  # Multiple ranges can be provided, each separated by a space.
  # Example: 192.168.1.1/32 192.168.2.1/32
  # The default set of ranges are those defined in RFC 1918 and typically used
  # within kubernetes clusters.
  IPRange: 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
```

# SLATE Command Line Interface

```
# Find the PerfSONAR testpoint application
$ slate app list | grep 'Name\|perfsonar'
Name App Version Chart Version Description
perfsonar-testpoint 4.2.0 1.0.3 Perfsonar Testpoint Deployment
# Get the default configuration
$ slate app get-conf perfsonar-testpoint > ps.yaml
# Customize the configuration
$ vi ps.yaml
# Do the install
$ ./slate app install perfsonar-testpoint --cluster uchicago-prod --group slate-dev --conf ps.yaml
Successfully installed application perfsonar-testpoint as instance slate-dev-perfsonar-testpoint-cnw- test
with ID instance_U-2KiIGqFKs
# Query instance information
$ ./slate instance info instance_U-2KiIGqFKs
Name                         Started                  Group     Cluster       ID
perfsonar-testpoint-cnw-test 2019-Jul-15 18:06:39 UTC slate-dev uchicago-prod instance_U-2KiIGqFKs
Pods:
  slate-dev-perfsonar-testpoint-cnw-test-84596d7c85-ns8xk
    Status: Running
    Created: 2019-07-15T18:06:44Z
    Host: sl-uc-xcache1.slateci.io
    Host IP: 192.170.227.137
# Run a test against the new endpoint
$ pscheduler task rtt --dest 192.170.227.137
Waiting for result...
1       192.170.227.137  64 Bytes  TTL 64  RTT   0.2690 ms
...
0% Packet Loss  RTT Min/Mean/Max/StdDev = 0.117000/0.190000/0.269000/0.051000 ms
```

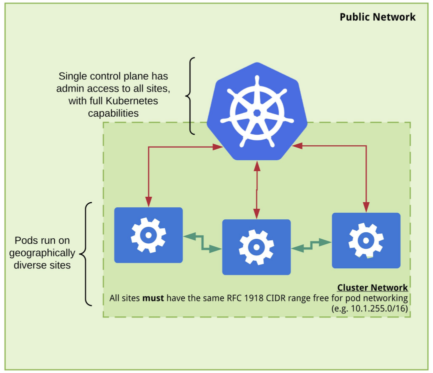
# Approaches to Kubernetes Federation

- Native federation (KubeFed) is still not mature (in alpha testing as of July 2019)
- 'Stretched' Kubernetes clusters become unwieldy at large scales, and have implications for networking
- For SLATE, giving users direct **`kubectl`** access to participating clusters was not a specific goal (and restricting what users can do is much easier without it)



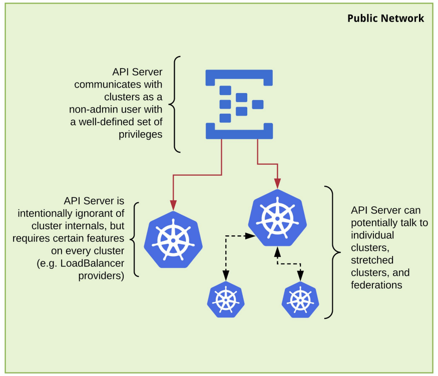
Public Network
Federation control plane is a special cluster that has admin privileges on all sites, with restricted Kubernetes capabilities
Sites run their own Kubernetes clusters that do not communicate directly across WAN
Cluster Network
Cluster Network
Sites free to choose whatever RFC1918 CIDR range is available

FEDERATION CONTROL PLANE



Public Network
Single control plane has admin access to all sites, with full Kubernetes capabilities
Pods run on geographically diverse sites
Cluster Network
All sites must have the same RFC 1918 CIDR range free for pod networking (e.g. 10.1.255.0/16)

STRETCHED KUBERNETES (a la PRP)



Public Network
API Server communicates with clusters as a non-admin user with a well-defined set of privileges
API Server is intentionally ignorant of cluster internals, but requires certain features on every cluster (e.g. LoadBalancer providers)
API Server can potentially talk to individual clusters, stretched clusters, and federations

SLATE PSEUDO-FEDERATION