EPJ Web of Conferences **245**, 07024 (2020) https://doi.org/10.1051/epjconf/202024507024
CHEP 2019

Towards a NoOps Model for WLCG

Robert Gardner[1,*] *Lincoln* Bryant[1], *Shawn* McKee[2], *Judith* Stephen[1], *Ilija* Vukotic[1], *Christopher* Weaver[1], and *Wenjing* Wu[2]

[1]Enrico Fermi Institute, University of Chicago, Chicago, IL, USA
[2]Physics Department, University of Michigan, Ann Arbor, MI, USA

Abstract. One of the most costly factors in providing a global computing infrastructure such as the WLCG is the human effort in deployment, integration, and operation of the distributed services supporting collaborative computing, data sharing and delivery, and analysis of extreme scale datasets. Furthermore, the time required to roll out global software updates, introduce new service components, or prototype novel systems requiring coordinated deployments across multiple facilities is often increased by communication latencies, staff availability, and in many cases expertise required for operations of bespoke services. While the WLCG (and distributed systems implemented throughout HEP) is a global service platform, it lacks the capability and flexibility of a modern platform-as-a-service including continuous integration/continuous delivery (CI/CD) methods, development-operations capabilities (DevOps, where developers assume a more direct role in the actual production infrastructure), and automation. Most importantly, tooling which reduces required training, bespoke service expertise, and the operational effort throughout the infrastructure, most notably at the resource endpoints (*sites*), is entirely absent in the current model. In this paper, we explore ideas and questions around potential *NoOps* models in this context: what is realistic given organizational policies and constraints? How should operational responsibility be organized across teams and facilities? What are the technical gaps? What are the social and cybersecurity challenges? Conversely what advantages does a NoOps model deliver for innovation and for accelerating the pace of delivery of new services needed for the HL-LHC era? We will describe initial work along these lines in the context of providing a data delivery network supporting IRIS-HEP DOMA R&D.

1 Scale of Operations

By almost any metric, the Worldwide LHC Computing Grid [1] has been enormously successful delivering the processing capacity required by the LHC experiments. The WLCG has 170 sites integrated into a global production fabric capable of delivering billions of CPU-hours per year while transferring hundreds of petabytes between storage endpoints at the various Tiers for processing and analysis. The software and computing teams of the LHC collaborations have built highly capable distributed workload and data management systems to exploit the aggregated resources of the contributing facilities (Figure 1). The result has been prodigious numbers of public results and graduated Ph.D. students.

*e-mail: rwg@uchicago.edu

The LHC experiments organize distributed computing operations into teams, usually one for distributed data management and the other for distributed production and analysis operations [2]. These teams of experts are typically backed up by a second tier of support personnel (dedicated "shifters") distributed across continents to provide 24x7x365 operational coverage. The tasks include optimizing storage utilization, monitoring file transfers and data throughput performance, ensuring data quality (checks for missing and/or corrupted data both centrally and at all of the WLCG sites), testing configuration and performance of site storage services, and the generic functions of monitoring, reporting, trouble triaging, and facilitating communication between WLCG site administrators and the experts throughout the system.

Figure 1: At left: the WLCG infrastructure which, across its Tier0, Tier1, and Tier2 centers, provided the bulk of the computing capacity for LHC Run 1 and Run 2 processing, storage, simulation and analysis. At right, distributed computing system components (in this case ATLAS [3]) which are typical of the LHC experiments.

The central teams are crucial to the smooth operation of the overall system and thus require significant experience and training; as a result the positions are at risk for understaffing or single-points-of-expertise-failure. Not visible to the central teams are the supporting tasks of the WLCG site managers and system administrators who are responsible for efficient and reliable operation of infrastructure and services in order to meet their WLCG pledged MOU commitments, including for availability and reliability metrics. Their tasks include operation of the computing and storage systems and supporting infrastructure, deployment and operation of the job and storage service endpoints (compute and storage elements), checking consistency between storage and central data management catalogs, troubleshooting file transfer and job errors, responding to tickets and emails to regional support lists from the central operations and shift teams. Often this may require reporting misconfigured tasks which result in job failure, problematic user jobs, and other issues in the distributed system that may or may not have been triggered by a local site problem. Thus, while overall the sytem itself performs very well and has met the demands of scale and complexity of the LHC experiments thus far, there are signficant labor costs and some future risks. Services can be difficult for non-experts to operate, and the training for new staff with complex and non-standard services and distributed systems is significant and long. Moreover, new development and innovation is difficult given the static (or slowly changing) nature of the widely distributed service environment. Updates across the many sites and differing e-infrastructures must be externally coordinated, introducing time delays for the central development teams.

2 Federating the Edge

The point at which institutional computing resources meet the LHC experiments is the "edge", i.e. the facility edge network where service endpoints are typically deployed on ro-

bust hosts connected to high capacity local and wide area networks. These zones are naturally fully under the control and responsibility of the local site manager, including enforcement of site access and cybersecurity policies.

In the WLCG edge of today we have a diverse and rich set of services connecting local resources to the global platform: services for network diagnostics (e.g. PerfSONAR throughput and latency measuring hosts), data transfer (storage) endpoints ("DTNs"), compute elements to route jobs to local resource managers (local job schedulers), software & conditions data caches (CVMFS, Frontier-squid), data caches (e.g. XCache), and in the case of ATLAS, the Harvester edge service (specifically for leadership class HPC facilities). In future? We can probably expect novel data delivery (transformations, reformatting, decompression, etc.) and data caching services. But one could imagine "facility APIs" (implementing Infrastructure-as-a-Service, IaaS), platform APIs (PaaS), and other modern abstractions may begin to play increasingly prominent roles. Scalable analysis platforms for HEP, often implemented using cloud-native technologies such as Kubernetes, are under development. We might expect some of these systems to appear during Run 3. While WLCG sites are "federated" within the context of the VO-site trust relationship of the grid model, there is no capability convergence in the service layer across sites. Deployment and operations must happen locally, at each site independently, by multiple individuals. This naturally leads to heterogeneity in the presented software release and configuration, as well as unevenness in quality of service.

2.1 NoOps and DevOps

To reduce site level differences, provide uniformity of configuration and evenness of quality of service, and reduce development (innovation) times, it is useful to explore a "NoOps" (no –local– operations) model where the focus is on automation, reducing the manpower required of the WLCG site team to support all of the systems needed for grid operations. This obviously is unrealistic for the majority of (existing) services, and so its context here is meant to refer to specific services which permit the development/deployment teams to operate in this manner (usually simple stateless services). We illustrate this with the example depicted in Figure 2.

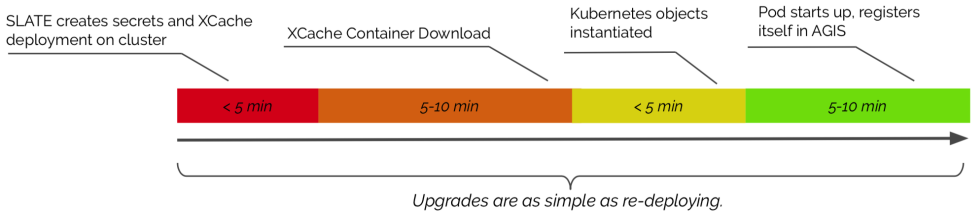

Figure 2: A deployment sequence for an XCache service using the Kubernetes-based SLATE federated edge platform.

In this example, an XCache service is deployed as a container image with a Helm chart configuration in a manner of minutes by an expert developer/operator. This can be scripted for several sites at once, permitting **regional or global** platform updates through a command line. The natural cloud-native analogy would be Netflix movie caches which, while located everywhere (in the network, on-prem at institutions), are DevOp'd centrally but NoOp'd locally [4]. There is no requirement of a local system adminstrator to fool with the cache when it breaks. Netflix is very good at fixing their caches, to the point of rehearsing breakage and repair, to prove to themselves they can (and measure the time required). Importantly, no local manpower is required to ensure a high quality of service for movie streaming to local clients.

2.2 Standardizing the edge substrate

A first step in this direction would be examine the typical service deployment arrangements found on WLCG sites. Note here we are confining the discussion to the edge, and only a subset at that. Currently, all of the infrastructure components are added using the preferred deployment and configuration management of the center (e.g. Puppet, Ansible, etc.) often on bare metal or a machine virtualization service such as OpenStack. This can allow for a mix of automation and declarative specification, depending on the chosen system. In many cases services are built "by hand". Essentially heterogeneity is introduced by differing hardware virtualization schemes, configuration management systems, and deployment practices across the sites.

By adding a consistent *edge substrate* layer, one that is common across facilities, and modular service components which use them, labor can be reduced and the odds of uniformity improved. This is schematically depicted in Figure 3. Depending on the technology used, there may be security challenges however, because instead of considering one service to permit (at a time), the site must consider the whole substrate. It is also obvious this would only become realistic if the software enjoyed broad community adoption, to the point of becoming a defacto standard (such as we have at the host image level with Linux).

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 itself deployed at each site may be managed centrally. This is broadly the approach taken by the Pacific Research Platform (PRP) [5] team, which operates an infrastructure which began in California but now extends internationally. Hardware may be controlled by local site admins, who then grant fine-grained permissions to external organizations for service operation. This is the approach we are discussing here. 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 (DOE labs, for example), have indicated they would require strict local controls. Services have traditionally only been the responsibility of the local administrator 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 the endpoints, and define how they will use resources.

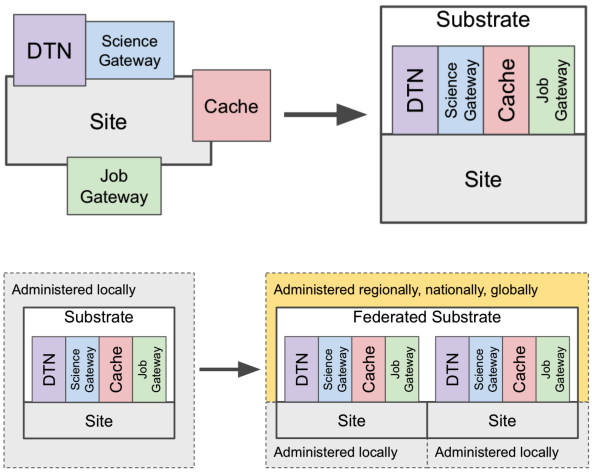

Figure 3: Top figure: individually installed edge services at a site can be deployed declaratively through use of a "substrate". Bottom figure: federated substrates can be built from sites, with services sharing common declarative deployment patterns on the individual sites. Note the partionining of administrative domains between the local NoOps and central DevOps teams.

3 SLATE Platform

The SLATE (Services Layer at the Edge) platform [6, 7] provides a substrate for this kind of approach. 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. It is designed to provide the necessary trust guarantees enabling service administrators to deploy and manage their software infrastructure effectively, while simultaneously maintaining the security of the computing resource providers' software, storage, computing, and network resources.

While SLATE uses Kubernetes [8, 9] as its underlying technology, the concept outlined here is largely independent of particular Kubernetes features, making it applicable to other implementations. SLATE supports both edge [10] infrastructure services, such as caches and data transfer endpoints, and domain applications which perform particular scientific calculations. While there are important differences between infrastructure services and domain applications, for the context of this paper they can generally be treated together and will be referred to generically as 'applications'.

3.1 SLATE Architecture

Many options are possible for federating multiple sites using Kubernetes, including "stretched" Kubernetes clusters with geographically distributed nodes, native Kubernetes federation [11], and add-on products such as Admiralty [12]. The approach taken by SLATE, as illustrated in Figure 4, is instead to introduce a central, custom component (described below), distinct from Kubernetes, to implement federation capabilities. This offers a number of advantages: a) a tailoring of the federation to the needs of the collaboration/experiment, b) a partial decoupling of the development of the SLATE platform from changes in the Kubernetes project itself, and c) allowing sites to retain full administrative control of their own Kubernetes edge cluster.

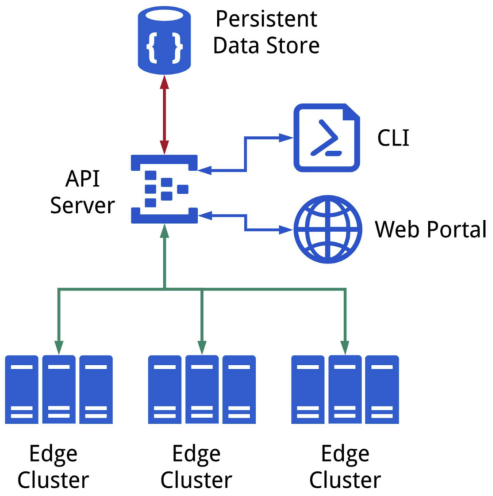

Figure 4: The central API server accepts user requests from the CLI and web portal, looks up data as needed from the persistent store, and generates commands to participating Kubernetes clusters.

The central SLATE component implements a REST API, accepts user requests, and forwards instructions to individual Kubernetes clusters as appropriate. Federated identity is based on InCommon and CI Logon, as implemented by the Globus Research Data Portal [13]. The state of the SLATE platform is persisted in a database, currently DynamoDB [14], which provides scalability and independence from hardware at a single site. When determining whether to forward instructions to Kubernetes, the API server accepts the user requests, validates the user with federated identity access, and applies authorization rules to determine whether the user in question has sufficient privilege for the requested action. Groups defined in the SLATE platform determine the capabilities of users. These same groups also authorize the user to

manage member clusters and application instances. Each resource provider which chooses to participate in the federation typically operates one Kubernetes cluster, on which multiple groups may deploy applications.

3.2 Application packaging

In order to deploy applications on the SLATE platform, developers containerize applications and package them through the use of Helm [15] charts and templates. Helm charts and templates are a simple format for providing some configuration values and for requesting Kubernetes object definitions. This approach makes application deployments simple and consistent. Charts are able to abstract application configuration settings out into a single document (in the form of a YAML [16] file). When the application is installed, the settings are substituted into the object definitions which are loaded into Kubernetes. Charts make usage simple for science users who are conversant with the configuration of the application (e.g. the amount of storage allocated for a data cache or the number of nodes to run in a distributed data analysis cluster) but may not be familiar with the syntax and structure of Kubernetes. Codifying applications into charts has the additional benefit that the chart represents a unified description of how the application will install. Another entity can then review and audit this unified description for several key security properties, such as code loaded and network ports utilized. SLATE will install only charts from specific catalogs to enforce a baseline level of trust for the sites. This is depicted in Figure 5.

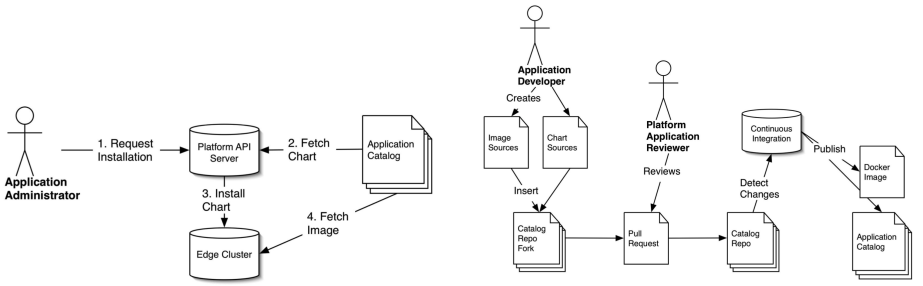

Figure 5: Left: Application install process on the SLATE platform. Applications are made available in a standardized catalog, with the SLATE API server enforcing that applications are installed only from this source. Right: Application review process for the SLATE platform. An application developer submits suggested changes, including both sources for a container image and a Helm chart, a platform reviewer considers the changes, and after approval by the reviewer an updated version of the application catalog is automatically published.

4 NoOps XCache Deployment

The goal was to build an XCache-based data caching network to support virtual data placement (VP) studies in the ATLAS production system. A number of SLATE-registered Kubernetes clusters became operational: the ATLAS Great Lakes Tier2 center (AGLT2) at the University of Michigan; the ATLAS Midwest Tier2 center (MWT2) at the University of Chicago; Střední Čechy, Česká republika Tier2 center (in Prague); and within the ESnet network (Sunnyvale PoP), which was deployed for a short duration test. An XCache application, customized for ATLAS, was curated into the SLATE application catalog. An ATLAS XCache DevOps team was organized.

The process involved deploying Kubernetes on a dedicated server at each site, thus creating a network of edge clusters. Each cluster was registered to the SLATE platform and configured to give the DevOps team (`atlas-xcache group`) deployment privileges. Additional, XCache-specific steps were followed: the node was labeled in Kubernetes (`xcache-capable=true`) to control placement of the service on the correct host; one or more storage volumes were mounted by the local system administrator and communicated to the DevOps team; and the XCache endpoint protocol was registered with the ATLAS information service.

The DevOps team created a test suit and containerized it. The team then could launch a realistic stress test from Google Compute Engine within minutes, to ensure a quick supply of clients to generate load. The DevOps team deployed all services from the command line interface. Notable benefits of the process included automatic XCache core dump collection (for central debugging of remote problems), and a simpler troubleshooting experience given the uniform container environment. Figure 6 shows sample deployment commands and Figure 7 gives the consolidated view of running XCache instances in the SLATE console, providing a convenient point of configuration, deployment state, and log data inspection.

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

Figure 6: SLATE command line deployment example illustrating deployment of an XCache service instance on a production Kubernetes cluster at the University of Chicago, within the MWT2 network and tested with the site's job queues.

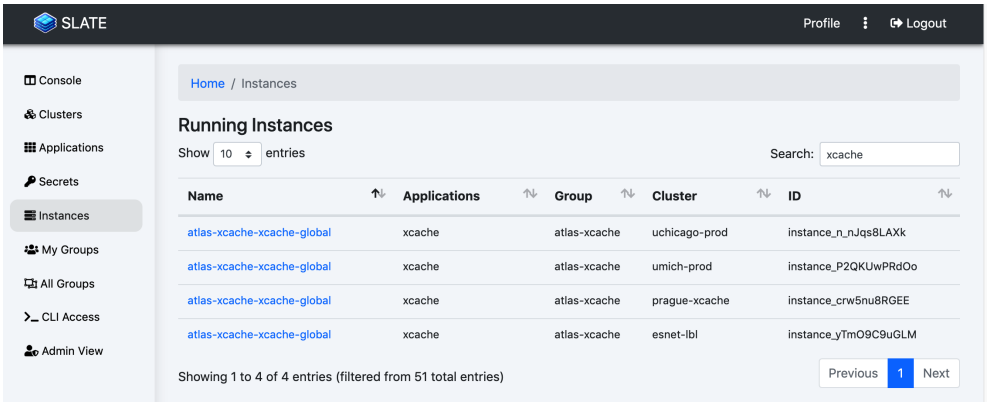

Figure 7: The SLATE console display of running XCache instances across the cluster network, with navigation links to detailed state and logging information for each of the deployed service instances.

5 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; that only appropriate software for

providing a relevant service will be run; that services will use appropriately secure software and configuration; that external users running services will not interfere with existing uses of resources.

5.1 Security and Policy Development

To address these and other concerns, the SLATE team underwent a software and security process review with TrustedCI [17, 18], the NSF Cybersecurity Center of Excellence. The major goal was to design security policies and procedures that address the issues of trust and privilege. 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 to 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 site concerns with respect to federated operation procedures and methods. And these documents must align with existing WLCG security policies.

5.2 WLCG Federated Operations Security Working Group

The deployment exercise of XCache with SLATE within the ESnet network first elevated the issue of container image security and lack of an established trust model for this approach to operations. This was due to vulnerabilities discovered in the application image and not the platform or the operational model. In response we solicited input from cybersecurity experts from the OSG and the WLCG, and ultimately helped form a federated operations security working group. A charter [19] was written at the pre-GDB meeting at Fermilab in September 2019. This was followed by discussion at the WISE workshop [20] in San Diego, co-located with the NSF Cybersecurity Summit for Large Facilities. The next step for the working group is to design a community survey to collect perspectives, concerns, and other information (e.g. institutional policies that might prevent adoption of a NoOps model). The survey will capture profile information of participants in order to examine responses according to location (country), e-infrastructure (grid organization), facility type (e.g. Tier), and professional role (systems administrator, site manager, cybersecurity official, etc.).

6 Summary and Future

Federated service operations across uniform substrates is possible today and is being advanced by a number of teams within trusted domains. For example, the U.S. ATLAS computing organization is deploying, operating and monitoring Xrootd and Frontier-Squid caching servers at their Tier-2 centers using SLATE and Kubernetes by a central team, accelerating the rollout of new features while reducing site administration effort. In addition to reducing operational effort at WLCG sites, the NoOps model can increase security, improve reliability, provide quicker updates, and allow central opertation by experts, potentially freeing local operational effort up for other tasks. A curated application catalog is an important component for trusted deployments and is one element of an overall federated operations security policy framework which must be developed for widespread adoption by the community. As WLCG computing sites explore the use of Kubernetes for infrastructure management generally, their application in edge networks provides a golden opportunity for federated service orchestration.

Acknowledgement

This work was supported by the National Science Foundation Office of Advanced Cyberinfrastructure grant OAC-1724821 (SLATE) and Cooperative Agreement OAC-1836650 (IRIS-HEP).

References

[1] I. Bird, P. Buncic, F. Carminati, M. Cattaneo, P. Clarke, I. Fisk, M. Girone, J. Harvey, B. Kersevan, P. Mato et al., Tech. Rep. CERN-LHCC-2014-014. LCG-TDR-002 (2014), https://cds.cern.ch/record/1695401

[2] Barreiro Magino, Fernando, Cameron, David, Di Girolamo, Alessandro, Filipcic, Andrej, Glushkov, Ivan, Legger, Federica, Maeno, Tadashi, Walker, Rodney, EPJ Web Conf. **214**, 03049 (2019)

[3] Elmsheuser, Johannes, Di Girolamo, Alessandro, EPJ Web Conf. **214**, 03010 (2019)

[4] *DevOps Case Study: Netflix and the Chaos Monkey*, https://insights.sei.cmu.edu/devops/2015/04/devops-case-study-netflix-and-the-chaos-monkey.html

[5] *Pacific Research Platform*, http://pacificresearchplatform.org/

[6] R. Gardner, J. Breen, L. Bryant, S. McKee, Science Gateways 2017 (2017)

[7] J. Breen, L. Bryant, G. Carcassi, J. Chen, R.W. Gardner, R. Harden, M. Izdimirski, R. Killen, B. Kulbertis, S. McKee et al., *Building the SLATE Platform*, in *Proceedings of the Practice and Experience on Advanced Research Computing* (2018), PEARC '18, pp. 5:1–5:7, ISBN 978-1-4503-6446-1, https://doi.org/10.1145/3219104.3219144

[8] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, ACM Queue **14**, 70 (2016)

[9] *Kubernetes | production-grade container orchestration* (2018), https://kubernetes.io/

[10] *Edge computing: Vision and challenges* (2019), https://ieeexplore.ieee.org/abstract/document/7488250

[11] *Kubernetes federation v2* (2018), https://github.com/kubernetes-sigs/federation-v2

[12] *Multi-cluster kubernetes. simplified. admiralty* (2019), https://admiralty.io

[13] K. Chard, E. Dart, I. Foster, D. Shifflett, S. Tuecke, J. Williams, PeerJ Computer Science **4**, e144 (2018)

[14] *Dynamodb* (2019), https://aws.amazon.com/dynamodb

[15] *The kubernetes package manager: Kubernetes helm* (2018), https://github.com/kubernetes/helm

[16] *Yaml ain't markup language* (2018), http://yaml.org/

[17] *Trusted CI begins engagement with SLATE*, https://blog.trustedci.org/2019/09/trusted-ci-begins-engagement-with-slate.html

[18] *Trusted CI Concludes SLATE Engagement*, https://blog.trustedci.org/2020/01/in-second-half-of-2019-trusted-ci-and.html

[19] *WLCG Federated Operations Security Working Group*, https://twiki.cern.ch/twiki/bin/view/LCG/WLCGFederatedOperationsSecurityWG

[20] *WISE as the NSF Cybersecurity Summit 2019*, https://wiki.geant.org/display/WISE/WISE+@+NSF+Cybersecurity+Summit+2019