SLATE

Lincoln Bryant
University of Chicago

GridPP42
24 Apr 2019

Motivation

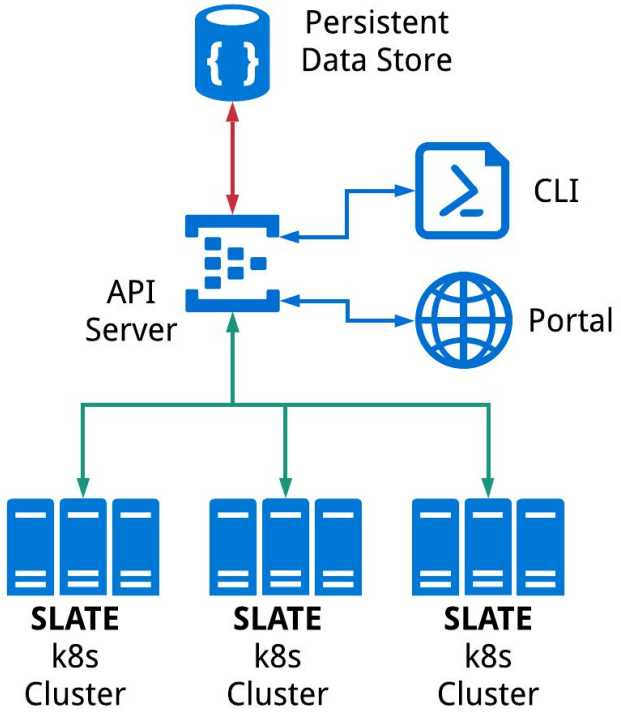
- Remotely manage edge services at sites with central expert teams
- Deploy updates more quickly
- Introduce new services more easily
- Save time and effort for local admins

Create a federation of edge platforms

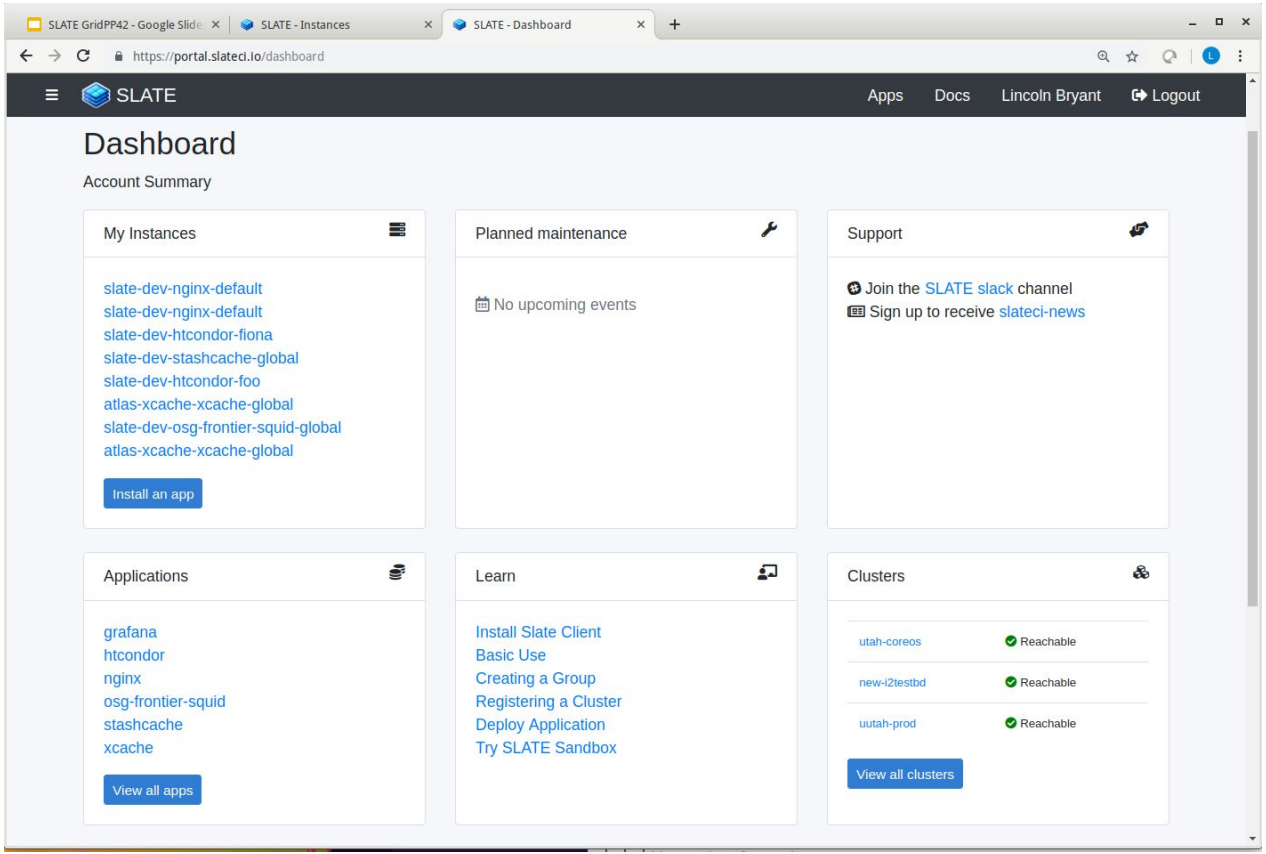
- **SLATE: S**ervices **L**ayer **A**t **T**he **E**dge
- Distributed service orchestration platform
- Loosely federated, share projects/users/applications across institutions
- Start with a single server and scale as needed
- Good for any site but "**lightweight**" sites might find it particularly useful

Basic SLATE Architecture

- Kubernetes-based platform for easily deploying, maintaining, and upgrading services across sites
- Federated via a custom server/client overlaying Kubernetes
- Software catalog with push button deployment
- RESTful interface with web portal and CLI client

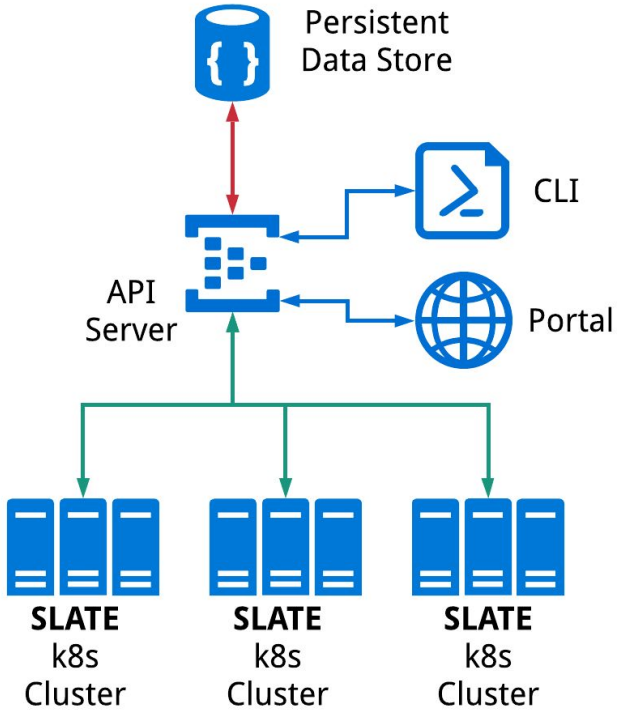


Web Portal


SLATE GridPP42 - Google Slide
SLATE - Instances
SLATE - Dashboard
https://portal.slateci.io/dashboard
SLATE
Apps
Docs
Lincoln Bryant
Logout
Dashboard
Account Summary
My Instances
slate-dev-nginx-default
slate-dev-nginx-default
slate-dev-htcondor-fiona
slate-dev-stashcache-global
slate-dev-htcondor-foo
atlas-xcache-xcache-global
slate-dev-osg-frontier-squid-global
atlas-xcache-xcache-global
Install an app
Planned maintenance
No upcoming events
Support
Join the SLATE slack channel
Sign up to receive slateci-news
Applications
grafana
htcondor
nginx
osg-frontier-squid
stashcache
xcache
View all apps
Learn
Install Slate Client
Basic Use
Creating a Group
Registering a Cluster
Deploy Application
Try SLATE Sandbox
Clusters
utah-coreos
Reachable
new-l2testbd
Reachable
uutah-prod
Reachable
View all clusters

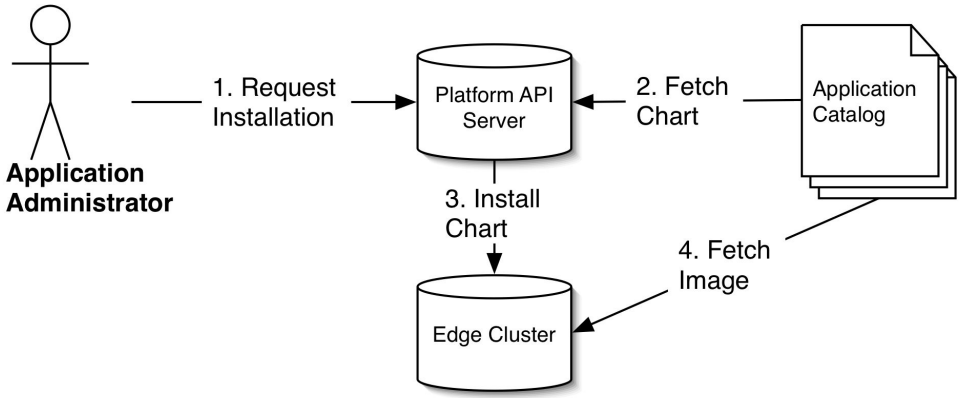
SLATE Lightweight Federation

- Security-conscious
  - SLATE platform is an unprivileged user of your cluster after first-time setup
  - http://slateci.io/blog/app-sec-edge.html
- Single entrypoint via institutional identity
  - Globus + InCommon
- Simple UNIX-like permissions model
  - Users, Groups
- Ability to whitelist groups and applications
  - e.g. group "atlas-xcache" can run XCache but not Squid
- Join the existing federation or build your own



Deploying Services with SLATE

- Curated application catalog on GitHub with Jenkins for continuous integration
- A central service expert deploys & operates the application for many sites
- Command line or web interface



Anatomy of a SLATE Application

- Docker container(s) with the software itself
- Templated YAML file describing the service in terms of Kubernetes objects (Pods, Ingress, Services, etc)
- Configuration parameter file exposed to the operator at deploy time
- Goes through an approval process with the SLATE catalog owners

Example Application: XCache

```
Service:
  Port: 1094
  ExternalIP: 192.170.227.151

SiteConfig:
  Name: MWT2
  AGISprotocolID: 433

XCacheConfig:
  CacheDirectories:
    - /scratch
  HighWaterMark: 0.95
  LowWaterMark: 0.90
  RamSize: 16g
  BlockSize: 1M
  Prefetch: 0
  CertSecret: xcache-cert-secret
```

Operator view, when deploying applications

120 lines (112 sloc) | 3.58 KB

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: {{ template "xcache.fullname" . }}
5    labels:
6      app: {{ template "xcache.name" . }}
7      chart: {{ template "xcache.chart" . }}
8      release: {{ .Release.Name }}
9      instance: {{ .Values.Instance }}
10 spec:
11   replicas: 1
12   selector:
13     matchLabels:
14       app: {{ template "xcache.name" . }}
15       chart: {{ template "xcache.chart" . }}
16       release: {{ .Release.Name }}
17       instance: {{ .Values.Instance }}
18
19   template:
20     metadata:
21       labels:
22         app: {{ template "xcache.name" . }}
23         chart: {{ template "xcache.chart" . }}
24         release: {{ .Release.Name }}
25         instance: {{ .Values.Instance }}
26     spec:
27
28       no
29
```

Templated K8S YAML, not directly exposed to operators

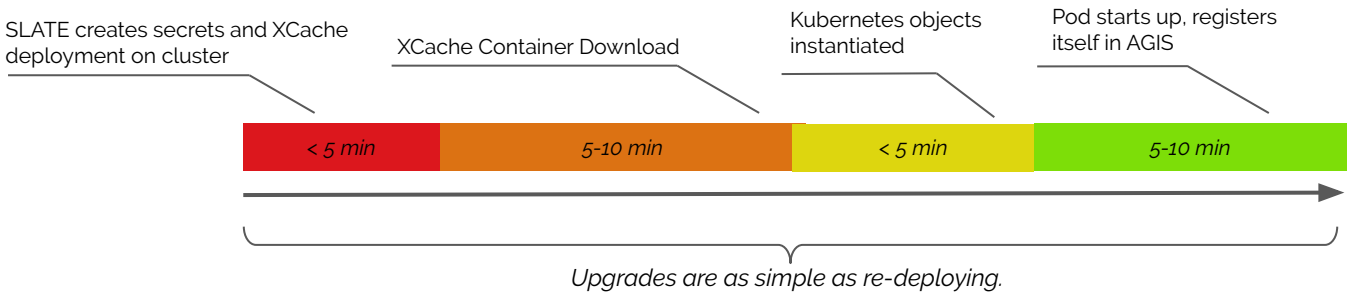
Deployment experience in ATLAS

- Goal: build an XCache-based caching network
- SLATE-based deployment will simplify operations and allow for rapid development and debugging
- SLATE services already operational at MWT2, AGLT2, LRZ
- XCache application already in SLATE catalog, frequently being iterated upon, updated at sites, and tested

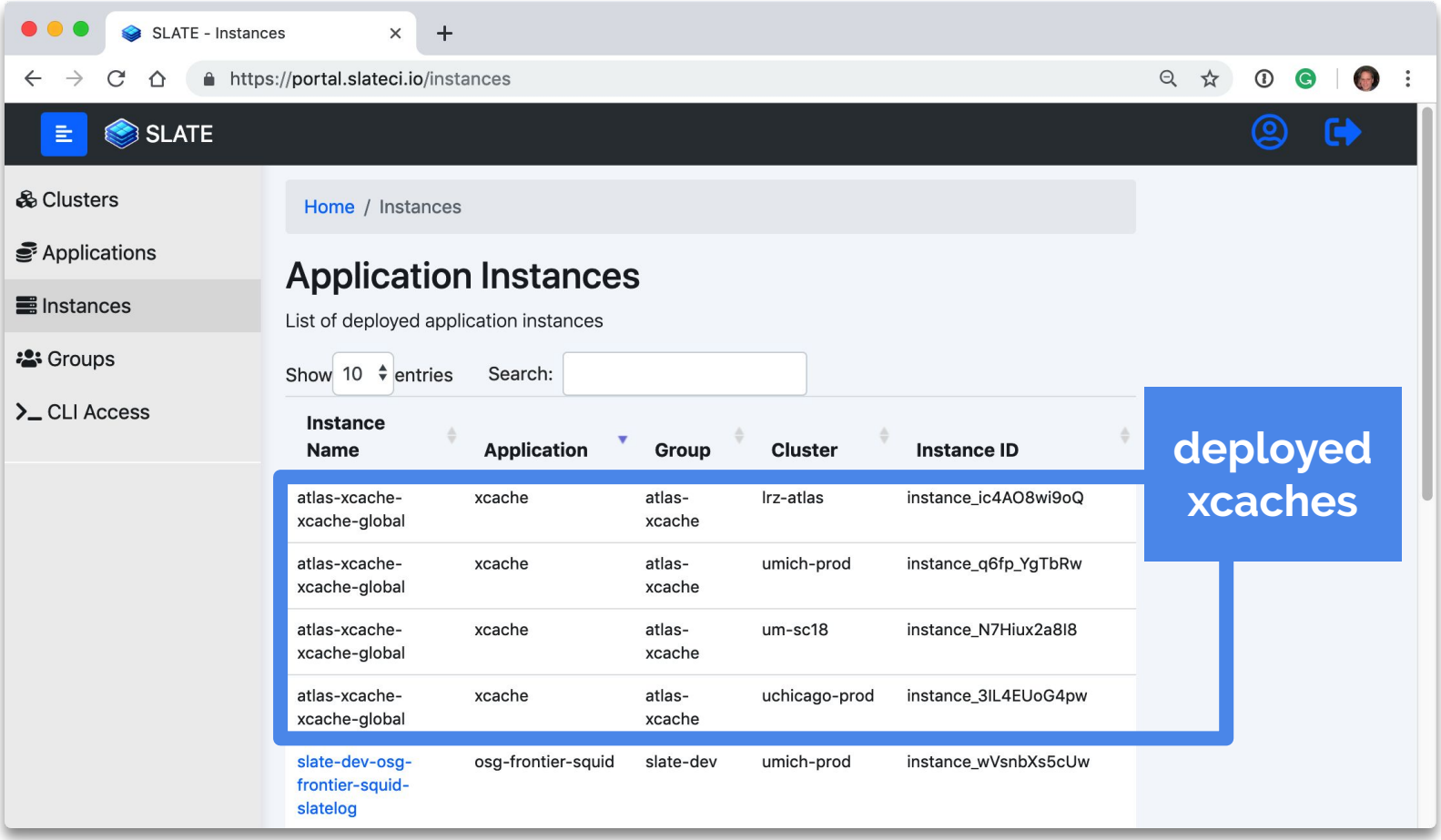
XCache deployment process

- 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 operators
  - Endpoint protocol registered in AGIS
- Test suite containerized
  - Launch a very realistic stress test from Google Compute Engine in minutes

XCache Deployment & Upgrade Cycle:



XCache fully deployed in less than 20 minutes.


SLATE - Instances
https://portal.slateci.io/instances
SLATE
Clusters
Applications
Instances
Groups
CLI Access
Home / Instances
Application Instances
List of deployed application instances
Show 10 entries
Search:
Instance Name
Application
Group
Cluster
Instance ID
atlas-xcache-xcache-global
xcache
atlas-xcache
lrz-atlas
instance_ic4AO8wi9oQ
atlas-xcache-xcache-global
xcache
atlas-xcache
umich-prod
instance_q6fp_YgTbRw
atlas-xcache-xcache-global
xcache
atlas-xcache
um-sc18
instance_N7Hiux2a8I8
atlas-xcache-xcache-global
xcache
atlas-xcache
uchicago-prod
instance_3IL4EUoG4pw
slate-dev-osg-frontier-squid-slatelog
osg-frontier-squid
slate-dev
umich-prod
instance_wVsnbXs5cUw
deployed xcaches

XCache updates

- 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 (part of XCache)
- Containerized environment makes it easier to debug

XCache Monitoring

Wealth of information collected even without any direct XRootD monitoring (summary or detailed stream).

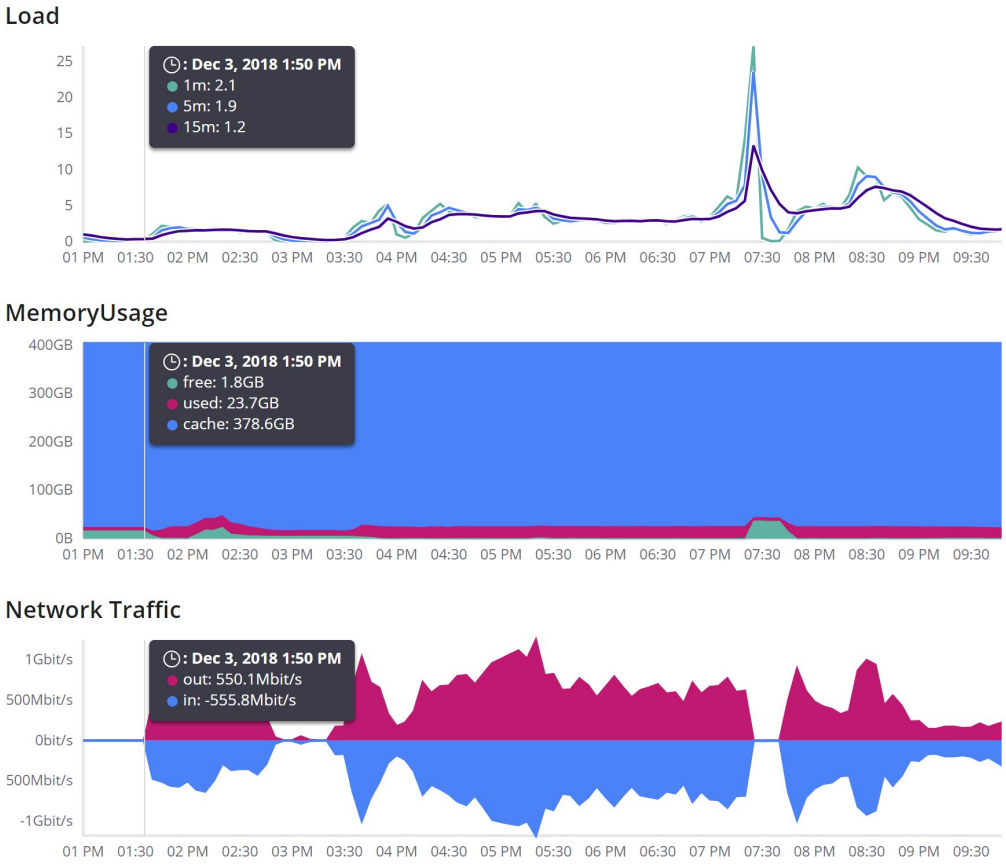
Node state (load, mem, network).

Per pod/container event and resource usage.

Logs. Fully searchable.

All info shipped to Elasticsearch at UChicago.

WIP - Prometheus-based monitoring



XCache Logging

Really convenient logging

- No need to contact anyone
- No need to log in anywhere
- Powerful search
- All the services logs in the same place.
- Kept for 10 days.

Infrastructure / Logs

BETA

kubernetes.pod.name: atlas-xcache-xcache-global-bdc76dbd9-pgqc4

Stream live

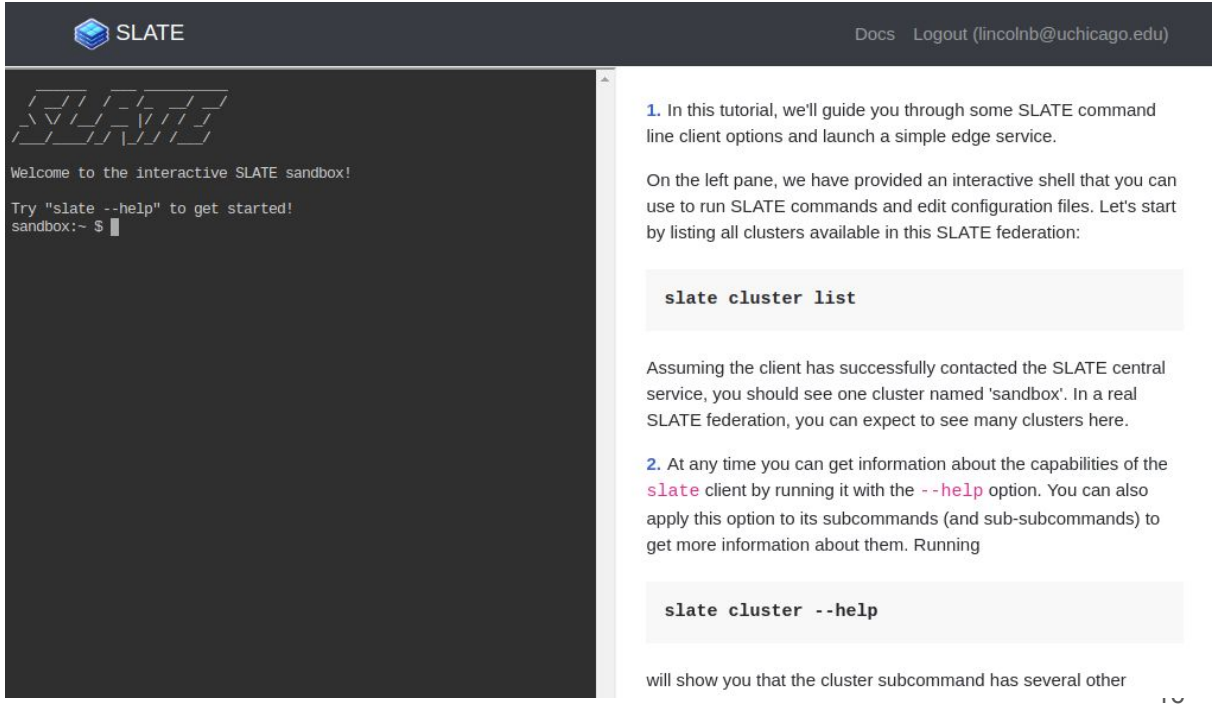
```
2019-03-02 11:02:25.908
2019-03-02 11:02:25.969  190302 10:02:25 5263 XrootdXeq: usatlas1.51
                         system.svc.cluster.local pub IPv4 login
2019-03-02 11:02:25.969  190302 10:02:25 5263 usatlas1.51852:261@192-170-227-156.prometheus-operator-kubelet.kube-system.svc.cluster.
                         ofs_open: 0-600
                         fn=/root:/xrootd.aglt2.org:1094/pnfs/aglt2.org/atlasdatadisk/rucio/mc15_13TeV/85/72/EVNT.06479607._035168.poo
2019-03-02 11:02:26.030  190302 10:02:26 5263 XrdFileCache_Manager: info Cache::Attach()
                         root://261@xrootd.aglt2.org:1094//atlas/rucio/mc15_13TeV/85/72/EVNT.06479607._035168.pool.root.1
2019-03-02 11:02:26.030  190302 10:02:26 5263 XrdFileCache_Manager: info Cache::Attach()
                         root://261@xrootd.aglt2.org:1094//atlas/rucio/mc15_13TeV/85/72/EVNT.06479607._035168.pool.root.1
2019-03-02 11:02:26.135  190302 10:02:26 5263 XrdFileCache_File: info ioActive block_map.size() = 0
                         /atlas/rucio/mc15_13TeV/85/72/EVNT.06479607._035168.pool.root.1
2019-03-02 11:02:26.135  190302 10:02:26 5263 usatlas1.51852:261@192-170-227-156.prometheus-operator-kubelet.kube-system.svc.cluster.
                         ofs_close: use=1
                         fn=/root:/xrootd.aglt2.org:1094/pnfs/aglt2.org/atlasdatadisk/rucio/mc15_13TeV/85/72/EVNT.06479607._035168.poo
2019-03-02 11:02:26.143  190302 10:02:26 5263 XrdFileCache_IO: info IOEntireFile::Detach() 0x856a9700
2019-03-02 11:02:26.144  190302 10:02:26 5263 usatlas1.51852:261@192-170-227-156.prometheus-operator-kubelet.kube-
                         system.svc.cluster.local ofs_close: use=0 fn=dummy
2019-03-02 11:02:26.145  190302 10:02:26 5263 XrootdXeq: usatlas1.51852:261@192-170-227-156.prometheus-operator-kubelet.kube-
                         system.svc.cluster.local disc 0:00:01
```

Sat 02
03 AM
06 AM
09 AM
12 PM
03 PM
06 PM
09 PM

Get a feel for it - SLATE "Sandbox"

- Starts a tutorial environment inside a Kubernetes pod with the SLATE client
  - Runs an instance of the SLATE API and exposes the cluster
- Anyone can make a temporary account, try out the command line interface, and deploy a simple web server application

https://sandbox.slateci.io:5000/



SLATE provisioning options

- **SLATE**Lite (for a quick evaluation using Docker):
  - https://github.com/slateci/slatelite
- Zero to k8s+**SLATE** script on a bare edge server:
  - Installs everything necessary starting from a fresh CentOS system
    http://jenkins.slateci.io/artifacts/scripts/install-slate.sh
- "Managed" install
  - **We** will SSH to your site, set it up, and hand you the configured machine.
- Full install
  - **You** install Kubernetes, download **SLATE** client and register your cluster

Registering a cluster

```
$ slate cluster create atlas-t2-xyz \
  --group atlas-xyz-admins \
  --org "ATLAS Tier 2 XYZ"
$ slate cluster allow-group atlas-xcache
```

- Join a kubernetes cluster to a SLATE federation
  - Specifying the group which will administer it and the organization which owns the resource
- Grant users access to deploy applications on the cluster
  - In this case, just the atlas-xcache group

SLATE Status & Roadmap

- Current release: v1.0
  - Variety of deployment methods (Shell scripts, SLATE Lite, manual instructions)
  - All Kubernetes objects needed for deploying our driving applications (XCache, Squid, HTCondor CE...) are available in the SLATE API and client
  - Portal functional for basic operations
  - Basic monitoring & logging infrastructure
- Next release:
  - Addressing persistent storage
  - Refactoring application deployment to use "ingress controllers" where applicable
  - Adding centralized DNS management component
  - Designing a user-friendly workflow from the portal
  - Expanded & polished monitoring and logging infrastructure

If you'd like to try out SLATE

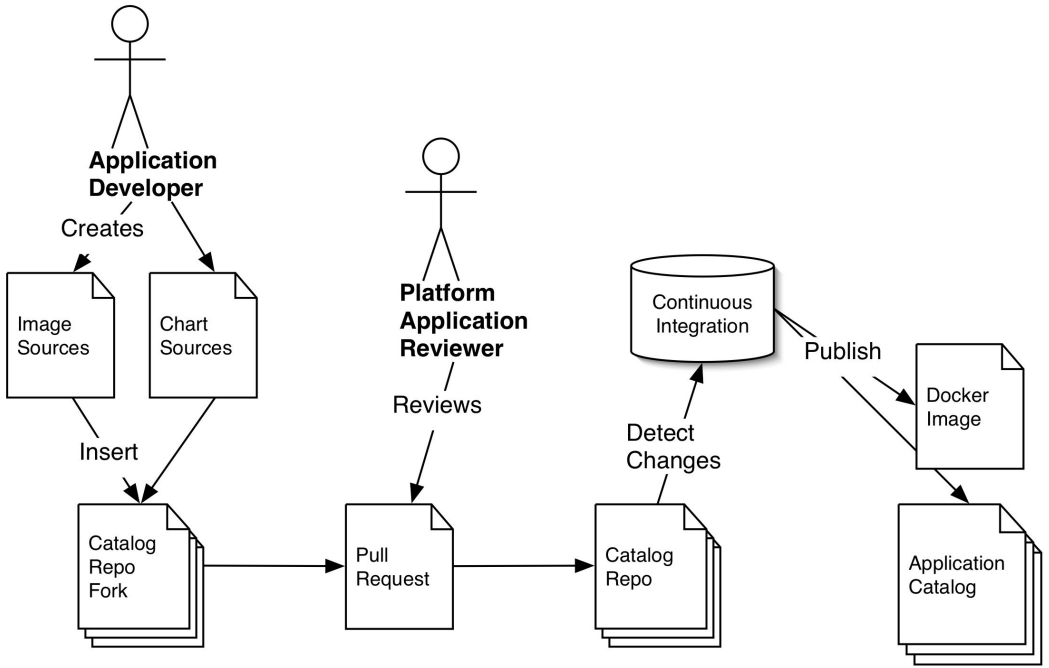
- Homepage: http://slateci.io/
- Slack: https://slack.slateci.io/
- Discussion list

Open to collaboration, packaging interesting applications, discussing use cases, etc!

Extra slides

Application Security for the Edge

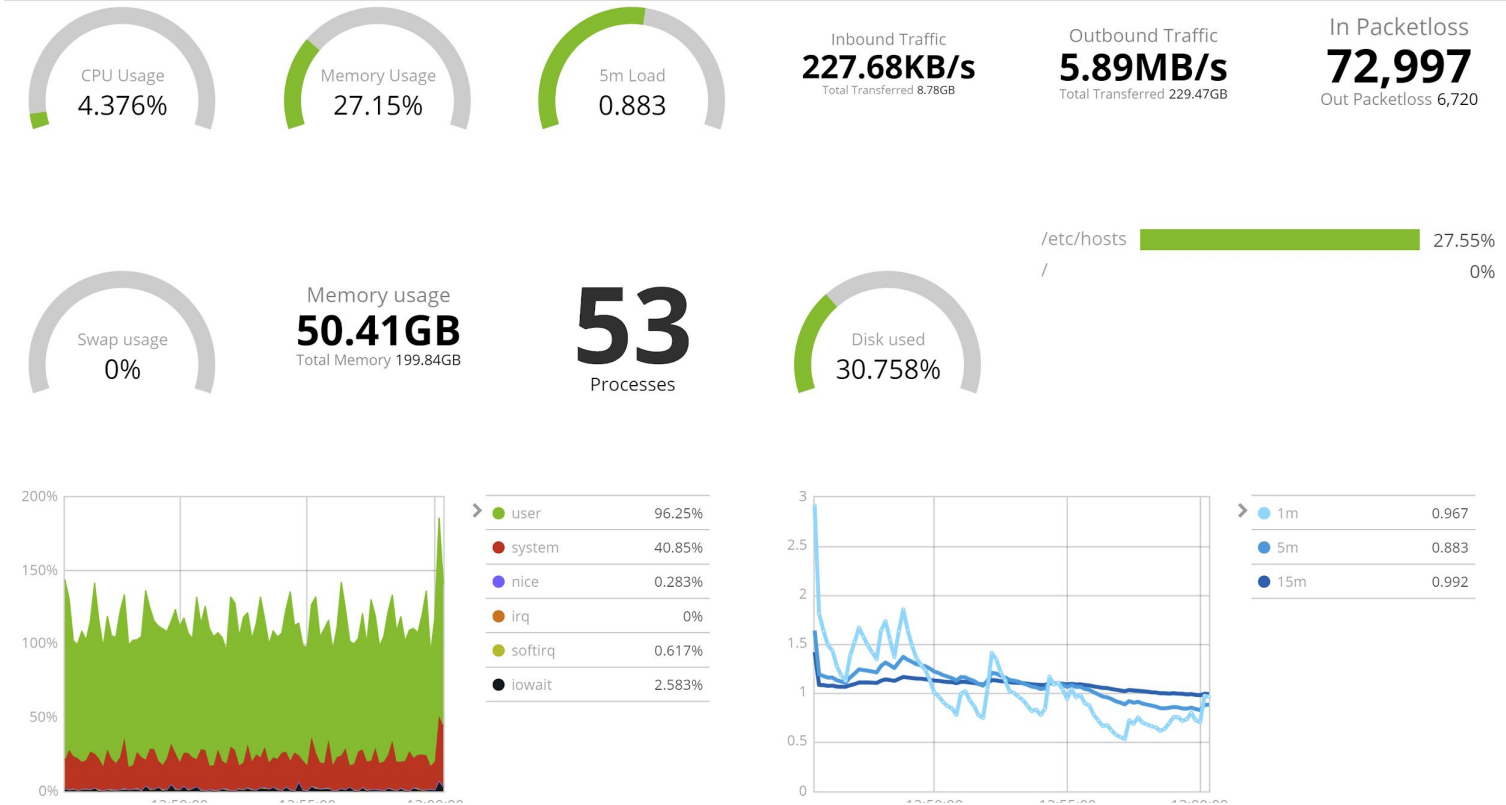
- We have considered the question of meeting sites' cybersecurity policies
- Discussions with community started: http://bit.ly/app-sec-edge
- Feel free to directly comment



XCache deployment process (more details)

- As XCache requires special resources this has to be communicated between Ilija and the site but is done only once:
  - Dedicated node labeled in K8s.
  - Storage should be JBODs organized.
  - Endpoint protocol registered in AGIS.
- Ilija takes over and creates secrets, server, reporting, monitoring, activates protocol in AGIS.
  - All of that is two commands and takes 30 seconds.
- Full update of all the caches in SLATE should take less than 20 min.
  - 10-15 minutes for dockerhub to rebuild image
  - 1 minute to stop running instances
  - 1 minute to start them again
  - 3 minutes to check everything worked
  - Even stress testing is containerized and Ilija can run a very realistic stress test against any xcache in matter of minutes (from Google Computing Engine)

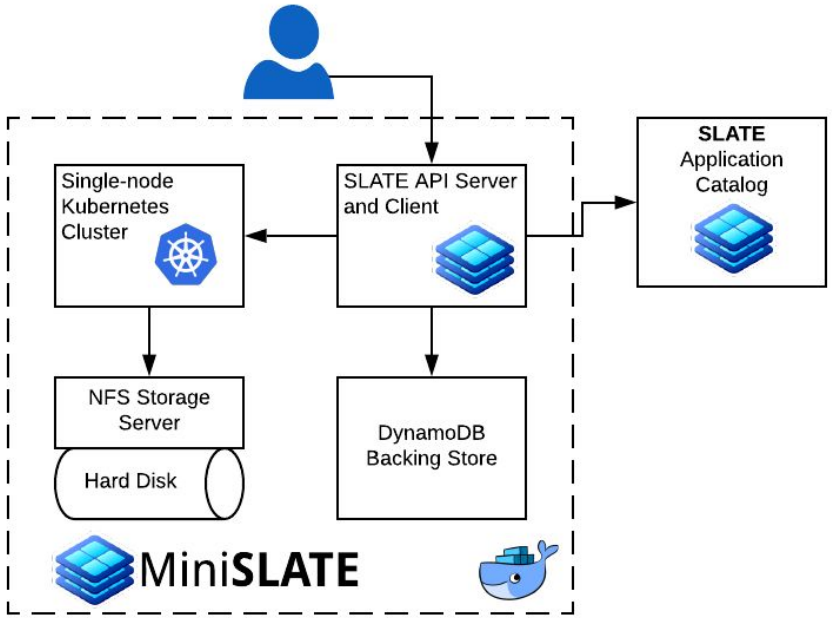
Monitoring - ES & Kibana



Mini**SLATE**

A development environment for SLATE

- **Create a stand alone, miniature SLATE federation for development**
- Follows an Infrastructure as Code pattern
- Enclosed within Docker
  - Little dependency clutter
    - Python, Docker, Docker-Compose
  - Environment consistency
- Completely Destructible
  - Destroy and recreate at will
  - Mount code from host safely
- Batteries Included
  - Full development kit
  - All required software and useful tools are installed when the Docker image is built



Installing Mini**SLATE** (https://github.com/slateci/minislate)

```
$ git clone https://github.com/slateci/minislate.git
Cloning into 'minislate'...
$ cd minislate
$ ./minislate init
(...)
DONE! MiniSLATE is now initialized.
$ ./minislate slate app install nginx --group ms-group --cluster ms-c

Installing application...
...
Successfully installed application nginx as instance ms-group-nginx-default with ID
instance_tey72YzGYuw
```

From last year 'til now

- Technology stack has firmed up significantly
  - **CentOS** as platform
  - **Kubernetes** for container orchestration
  - **Helm** for application packaging and deployment
  - Home-grown **SLATE API server, web portal and client**
- SLATE hardware in-place and configured at Chicago, Utah, Michigan
- External customers coming on board (e.g. LRZ)
- Application catalog best practices being developed