The Ubiquitous Edge Platform

Lincoln Bryant
Rob Gardner

WLCG GDB Meeting
May 10, 2016

Ubiquitous & Easy “CI Substrate”

- Pioneer a new phase of advanced cyberinfrastructure deployment, allowing sites to flexibly evolve and sustain both on-premise and commercial cloud-based infrastructure
- Hosted services, such as CEs, data caches, squid, etc., could be centrally deployed onto “CI substrates” within a trusted CI zones and remotely operated, upgraded, and optimized for performance
- Extend to shared, opportunistic university clusters and cloud resources

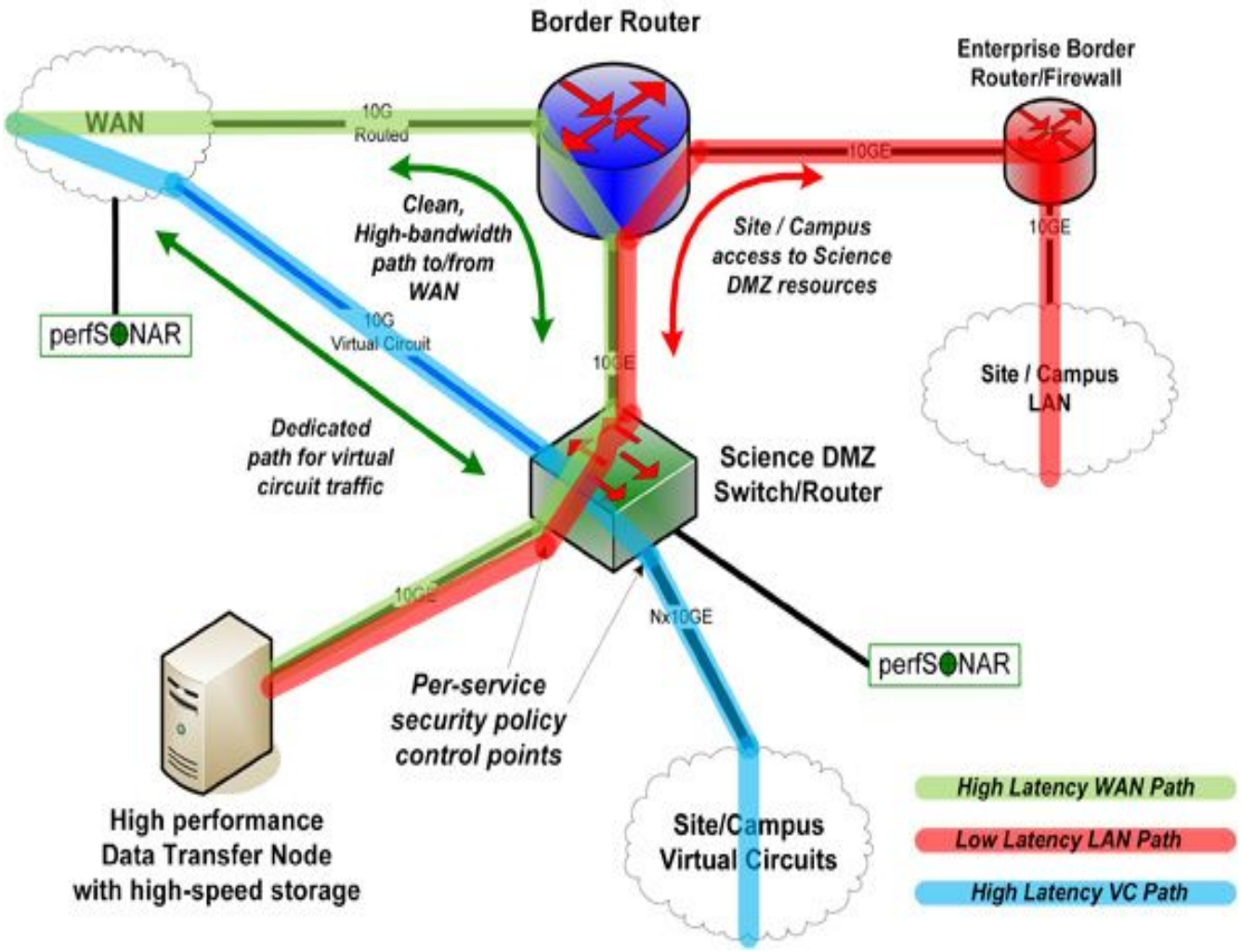
Motivation

- Coordination and operation efforts take a considerable amount of people time
  - Configuration, new versions, vulnerability patching, site-specific peculiarities
- Simplify site deployment across the WLCG
- Provide a framework for other science communities and future applications

Distributed Virtualized Data Centers

- Reduce IT footprint and ops burden
  - Centralize deployment & ops; reduce local admin cost
- Explore virtualized data center frameworks
  - E.g. container management over bare metal or VMs
- “Blue sky” goal
  - Establish a “trusted pattern” for a “CI substrate” on sites
  - Create distributed virtualized data center(s) overlaying the fabric substrate

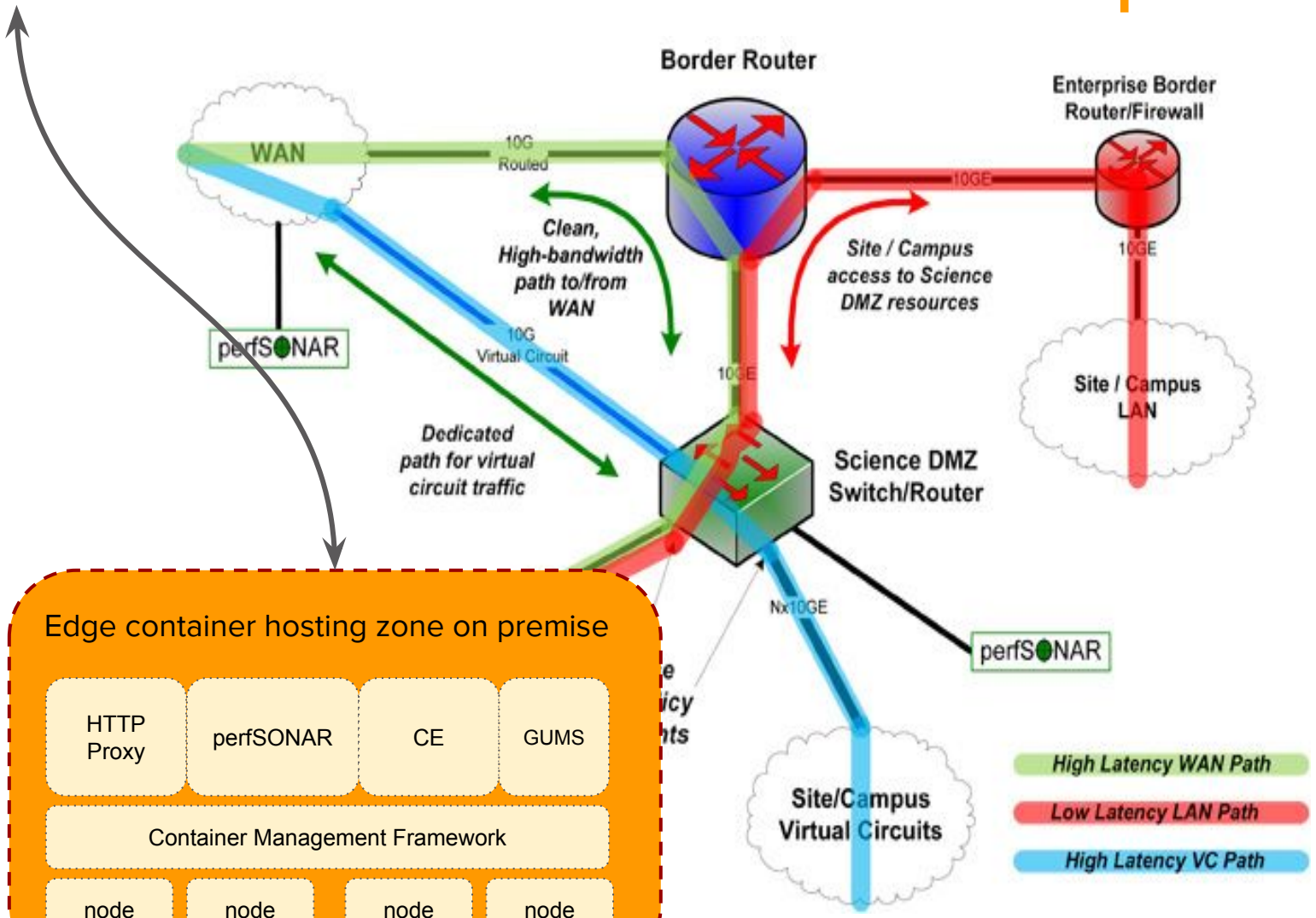
Canonical SciDMZ



CI central ops console:

$ slate install osg-squid.3.3 --sites SiteA SiteB SiteC

SuperDMZ



Edge container hosting zone on premise

HTTP Proxy	perfSONAR	CE	GUMS
Container Management Framework			
node	node	node	node

Deploying research software at the edge

Open Science Grid

perfSONAR

XRootD

globus

Hardware specification

- Produce a reference specification for edge container platform
- Allows support team to focus on software curation rather than hardware troubleshooting and optimization
- Initial focus on low-cost, single node deployment
- Could potentially be used to ‘seed’ a larger, multi-node infrastructure

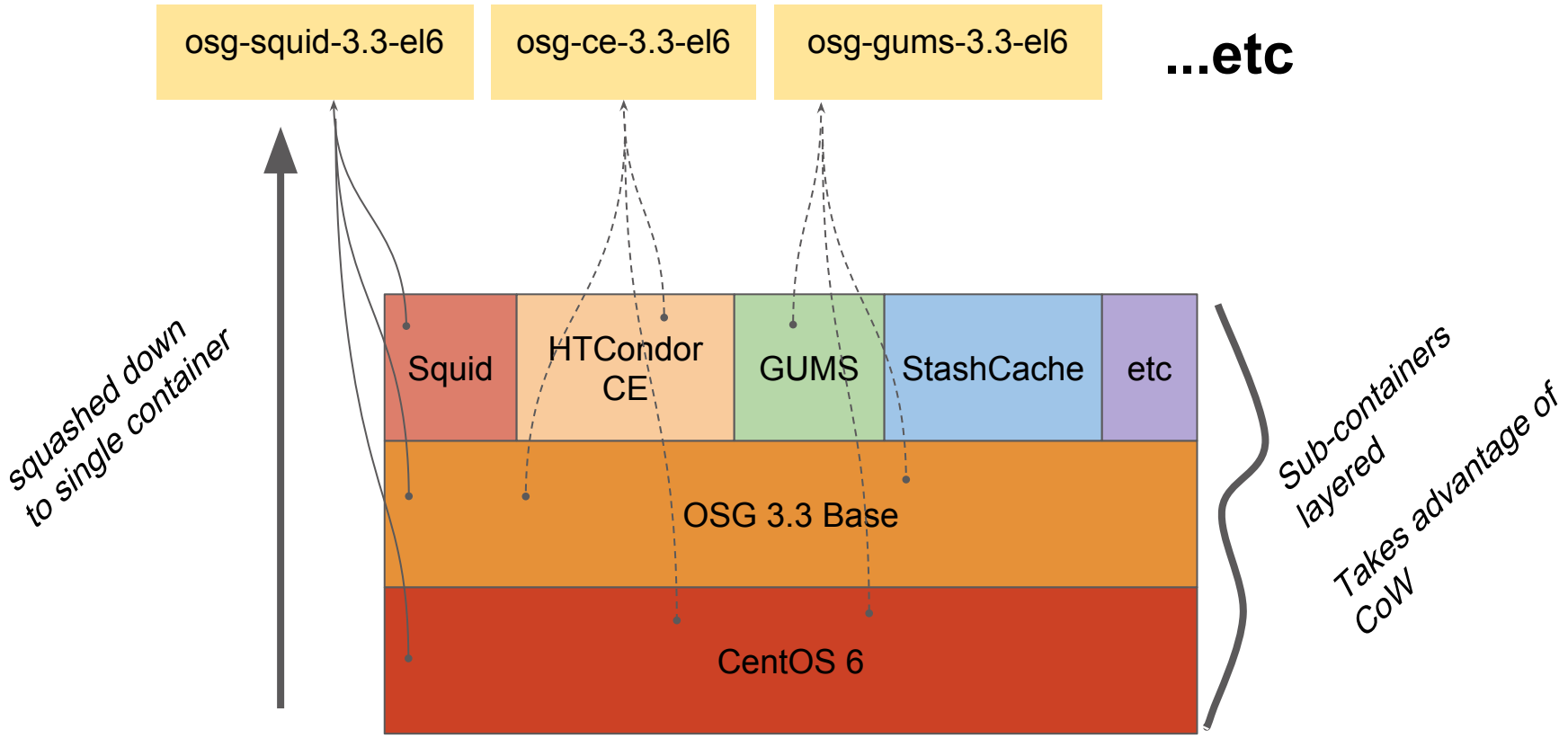
Software underpinning

- Exploring a number of options for container runtime and base operating system
  - Docker, LXD, CoreOS, SmartOS, etc
- Trying to find a happy medium between platform features and ease-of-use for local site administrators
- Web interface and RESTful API
  - Requires development work, out-of-the-box options are scarce

Application deployment

- Containerized applications created, vetted, maintained by central operations team.
  - Pushed by operators down to subscribed sites
  - Or, depending on use case, pulled by local site admins without interaction with central.
- Built-in monitoring/analytics
  - Every service should get its own set of applicable collectors
  - Leverage our existing monitoring expertise

Containerizing Services



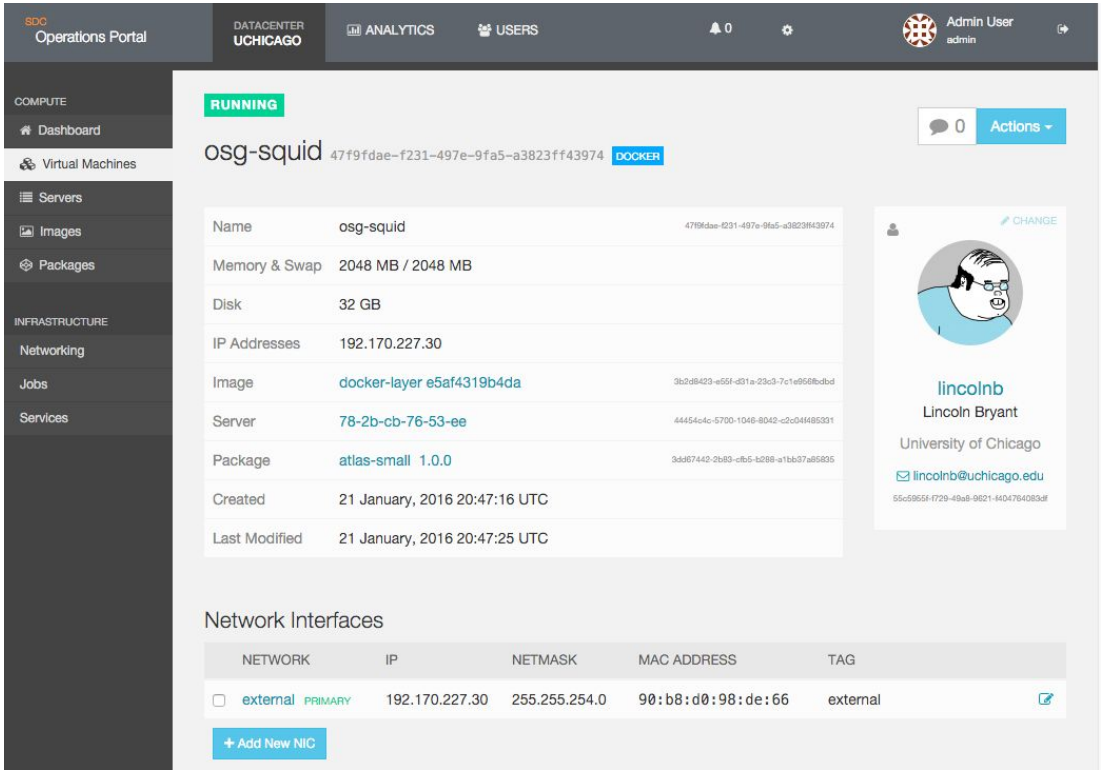
Example: Frontier Squid - Dockerfile

```
FROM lincolnbryant/osg-base-3.3-el6
MAINTAINER Lincoln Bryant <lincolnb@uchicago.edu>
# See https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallFrontierSquid

RUN yum install -y frontier-squid initscripts

VOLUME ["/var/cache/squid"]

COPY customize.sh /etc/squid/customize.sh
RUN chown squid: /etc/squid/customize.sh && chmod +x /etc/squid/customize.sh

EXPOSE 3128 3401

CMD /sbin/runuser -s /bin/bash squid /usr/sbin/fn-local-squid.sh start && tail -f
/var/log/squid/*.log
```

Example: Frontier Squid - Launching

- The container can be launched on another machine, or via Docker's remote API to a cloud resource

```
$ docker run -p 3128:3128/tcp -p 3401:
3401/udp -ti -e IP_BLOCKS="10.0.0.0/8
192.170.226.0/23" -e MEMORY_MB=2048 -e
CACHE_MB=32768 lincolnbryant/osg-squid-3.3-
el6

Generating /etc/squid/squid.conf

Initializing Cache...

2016/01/21 20:45:07| Creating Swap
Directories

Starting 1 Frontier Squid...

done


...
```



Automation efforts

- It should be possible for me to stand up and destroy a site in a completely automated way.
- Many points where human interaction is currently needed.
- Can we separate approvals (requiring human interaction) from configuration?
  - OIM, AGIS, and equivalents
  - Certificate registration
  - etc

Benefits for WLCG

- Could potentially deploy CEs, SEs, caching proxies, etc all within “the box”.
  - Best known versions and configurations get automatically pushed to downstream
  - Updates should be atomic, so rollbacks are easy.
- Containerization effort putting more eyes on existing documentation and builds
  - Example: Patches submitted for GUMS to build on EL7
    - https://github.com/opensciencegrid/gums/pull/27

Summary

- Platform for “edge” services on Science DMZs with well-defined reference hardware.
- Container-based applications, maintained by a central team
- Built-in service discovery, configuration, and monitoring
- Flexible, adaptable to the needs of other projects.

Thank you!
Questions?

Extra slides / Open questions

Security concerns

- Who has root on the machine?
- Can trusted users allocate resources and start containers remotely?
- Is Docker secure enough to be used? Many claims of a busted security model.
  - User namespaces and unprivileged containers seem to be semi-working in new kernels? (Affects OS choice!)
- Ultimately: What is the correct privilege separation between owner and operator?

Other considerations

- What does the networking configuration look like?
  - Do standard installers cover the majority of network configurations for initial bootstrapping?
  - Private control channel / VPN?
  - Require public IP(s)?
- Can we use this platform as a testbed for things like SDN?
- What does it look like when we have multiple nodes per site?