Developing for a Services Layer at the Edge

Ben Kulbertis
University of Utah

HEPiX 2019 Workshop
March 25-29, 2019

SLATE: Services **L**ayer **A**t **T**he **E**dge

- NSF DIBBS project (UChicago, UMichigan, UUtah)
- Distributed service orchestration platform
- Kubernetes-based
- Start with a single server and scale as needed
- Share projects/users/applications across institutions

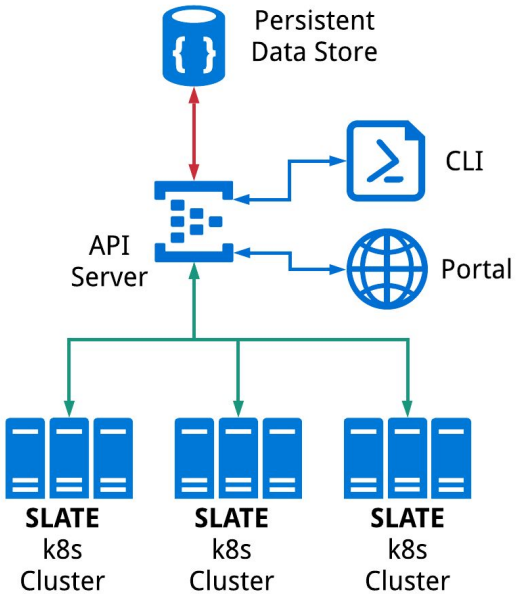
Goal

- 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

Accelerate creation of multi-institution research platforms

Basic SLATE Architecture

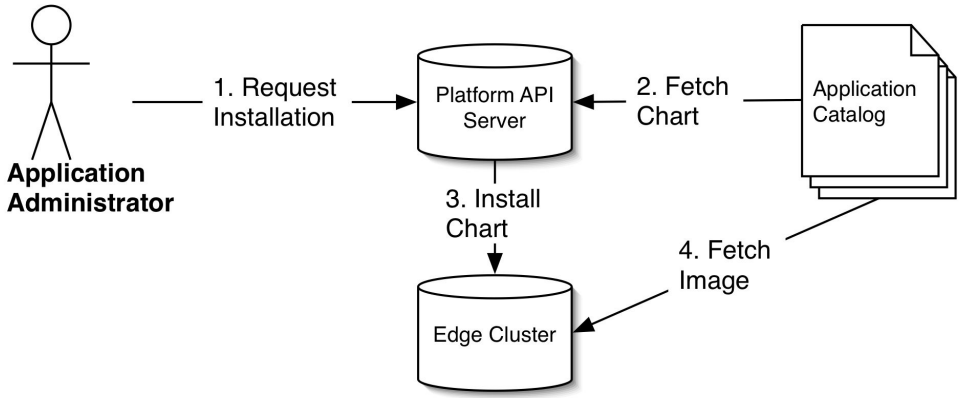
- Lightweight federation and application catalog layer on top of Kubernetes
  - Security-conscious, site autonomous
  - Sites retain administrative control
- Single entrypoint using institutional identity
- Simple UNIX-like permissions model (Users + Groups)
- **Application catalog** provides natural boundary between configuration knobs users actually want to change and complex Kubernetes configurations
- SLATE is an infrastructure **and software**



Create & manage your own federation over independently managed Kubernetes clusters

Deploying Services ("Applications" in k8s)

- A "central" service expert deploys & operates many sites
- Helm charts and Docker images
- Command line or web interface (in dev)



Application Catalog

- Contains all applications that are packaged to be installed on the SLATE platform.
- Based on Helm Charts - https://helm.sh
- Developers can create Helm Charts for the SLATE platform and submit them to the catalog.
- Incubator and Stable Repositories
- Applications are manually reviewed if not from known and trusted sources/organizations.
- Container image security policy

Why Helm?

- Kubernetes is complex
  - Application developers write once for users
  - End-users require less deep Kubernetes knowledge
- Environments are different
  - Take advantage of templating for configuration variables
  - Developers need not worry about exact deployment details
- Package management
  - Keep a curated catalog of charts
  - “Push button” deployment and deletion of apps
- This results in improved productivity
  - Improved efficiency for core and application developers

Helm Charts

Chart.yaml

```
# Version of Kubernetes in use
apiVersion: v1
# Version of application packaged for
installation
appVersion: 3.5.27
description: A Helm chart for configuration
and deployment of the Open Science Grid's
Frontier Squid application.
name: osg-frontier-squid
# Chart version
version: 1.0.0
```

Values.yaml

```
Instance: global
SLATE:
  Logging:
    Enabled: true
    Server:
      Name: atlas-kibana.mwt2.org
      Port: 9200
  Cluster:
    Name: ms-c
  LocalStorage: false
Service:
  Port: 3128
  ExternalVisibility: NodePort
SquidConf:
  CacheMem: 128
  CacheSize: 10000
  IPRange: 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
```

Dedicated Development Environments

Pros

- Little~No setup for developers
- Little sysadmin experience required for developers, even for advanced configs
- Consistent for all developers

Cons

- Consistency issues with production (especially without IaC)
- Volatility (single bad deploy brings down environment for all developers)
- Maintenance (OS management, “refreshes”, etc.)
- Security / IAM requirements
- Requires dedicated resources

Local Development Environments

Pros

- Limited volatility (one developer, one environment)
- No maintenance (environments are codified and destructible)
- Developer flexibility (reset environment at will, modify environment as needed)
- Ease of use (less security barriers, no need to push code remotely)

Cons

- Environment variability (high configuration ability required)
- Limited resources (software must run on local machines)
- Machine clutter (personal machine can easily become a dedicated development machine when many dependencies are required)

The Ideal: Best of Both Worlds

- Little setup for developers
- Little sysadmin experience required for developers
- Consistent for all developers
- Limited volatility (one developer per environment)
- No maintenance (environments are codified and destructible)
- Flexible (reset environment at will, modify environment as needed)
- Easy to use (few security barriers, no need to push code remotely)
- Runs wherever developers want to develop

Minikube

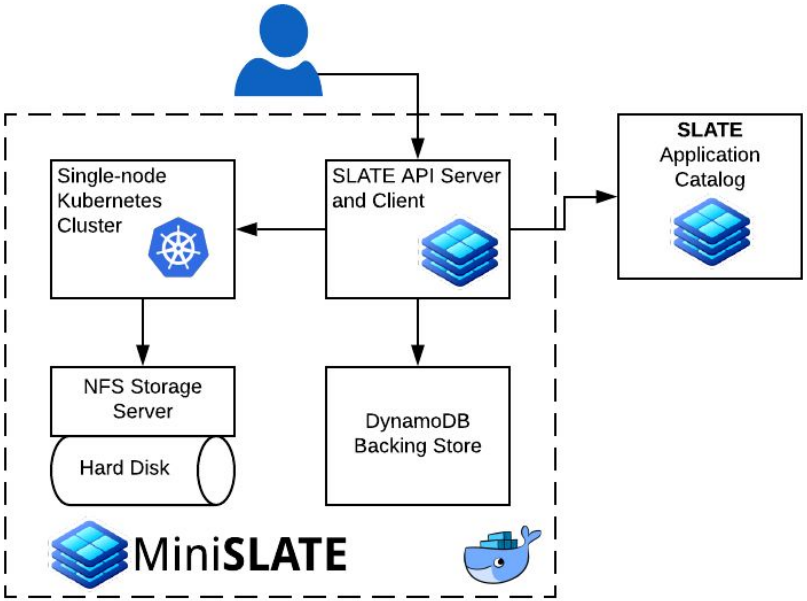
- Most popular development tool for Kubernetes
- Limited configuration ability for kubeadm
- Uses a VM, creating some system overhead
- Requires kubectl, slate client, on host
- Storage limitations
- **Can't deploy exact same chart code on production cluster**

MiniSLATE

A development environment for SLATE

https://github.com/slateci/minislate

- **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



Kubernetes: Containerized

- MiniSLATE utilizes an architecture similar to https://github.com/kubernetes-sigs/kubeadm-dind-cluster
- Kubernetes is contained within a Docker container, allowing it to be utilized as a standalone service with little host interference.
- This k8s instance can be utilized from the host or from another networked container. MiniSLATE packages an entire SLATE environment within a networked container and utilizes this k8s instance, creating a full SLATE deployment on your laptop.
- All SLATE and k8s dependencies are containerized, reducing clutter on the host system.

Installing MiniSLATE

```
$ git clone https://github.com/slateci/minislate.git
Cloning into 'minislate'...
$ cd minislate
$ ./minislate init
(...)
Default Group: ms-group
Default Cluster: ms-c

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
```

MiniSLATE CLI Interfaces

Command Wrapper

```
$ ./minislate slate instance list
Name          Group          Cluster ID
nginx-default ms-group ms-c  instance_fXlxlZPtkGY

$ ./minislate slate app install /mnt/mychart
Installing application...
Successfully installed application mychart as instance
ms-group-chart-default with ID instance_fXlxlZPtkGY

$ ./minislate shell slate
# whoami
root
# slate instance list
Name            Group           Cluster ID
Nginx-default   ms-group ms-c   instance_fXlxlZPtkGY
Mychart-default ms-group ms-c   instance_K1c8fjXpq1T
```

Aliases

```
$ source shell_aliases
$ cd ~
$ slate instance list
Name          Group          Cluster ID
nginx-default ms-group ms-c  instance_fXlxlZPtkGY

$ kubectl get nodes
NAME           STATUS   ROLES    AGE   VERSION
e0934ffdc141   Ready    master   11m   v1.14.0

$ minislate destroy -y
Killing minislate_kube_1 ... done
Removing minislate_slate_1 ... done
Removing minislate_kube_1  ... done
Removing minislate_db_1    ... done
Removing minislate_nfs_1   ... done
```

Publishing Charts

1. Submit your chart to the SLATE application catalog with a pull request to the catalog repository.
2. Initial publications should be placed in ‘incubator’, and once tested will be moved to ‘stable’.
3. Charts will be reviewed for compliance with the SLATE image security policy.
4. Charts in the stable repository can be deployed at any SLATE site.

More Info

- Slides: https://bit.ly/slate-hepix-2019
- MiniSLATE: https://github.com/slateci/minislate
- SLATElite: https://github.com/slateci/slatelite
- Homepage: http://slateci.io
- Slack: http://bit.ly/slate-slack-03
- Discussion list

Thank You

Acknowledgements

- University of Chicago
  - Lincoln Bryant
  - Chris Weaver
  - Rob Gardner
- University of Utah
  - Jason Stidd
  - Joe Breen
- University of Michigan
  - Shawn McKee

National Science Foundation CIF21 DIBBs: EI: SLATE and the Mobility of Capability, Grant No. 1724821.