

# 2019 TECHNOLOGY exchange

DECEMBER 9-12  NEW ORLEANS LA

## Running the InCommon Trusted Access Platform in the Cloud

“Look Ma, No Servers!”

### PRESENTER NAME:

**Keith Wessel** University of Illinois - Urbana-Champaign  
**Ethan Kromhout** University of North Carolina - Chapel Hill  
**Erik Coleman** University of Illinois - Urbana-Champaign  
**William Thompson** Lafayette College  
**Chris Hyzer** University of Pennsylvania  
**Christopher Hubing** Internet2



# Trusted Access Platform on GKE

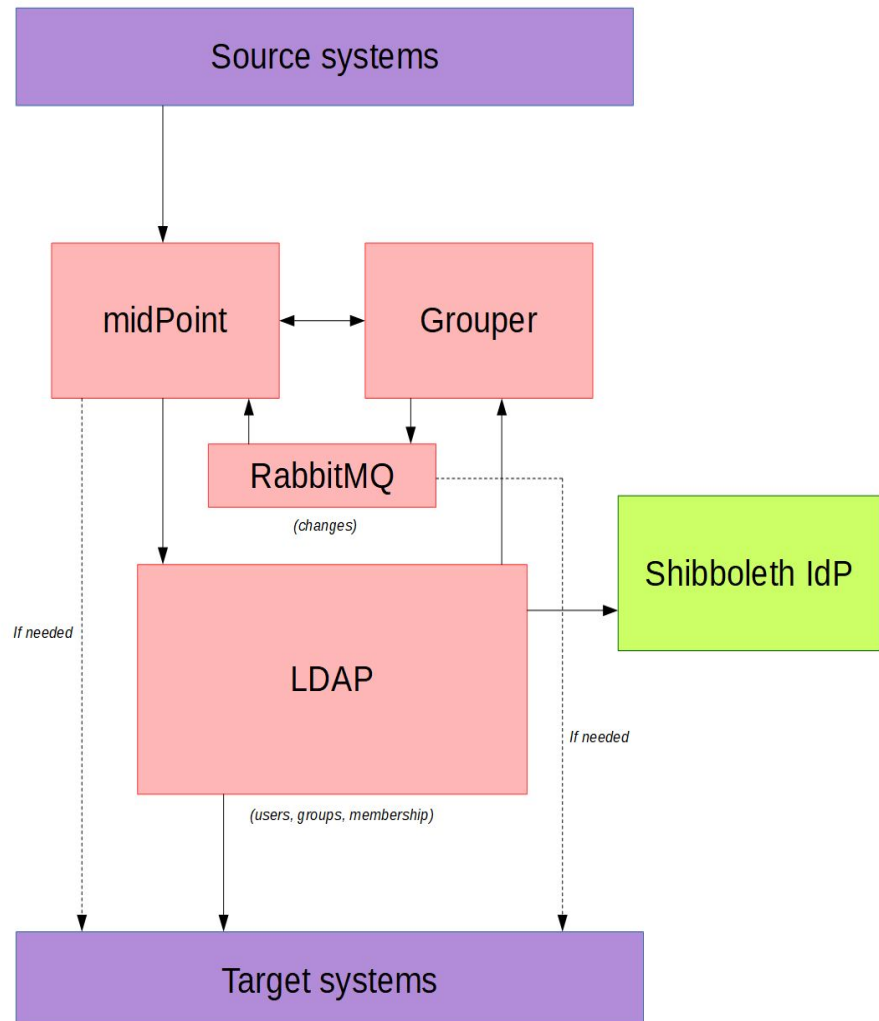
## Migrating a docker-compose built demonstration environment into Google Kubernetes

Ethan Kromhout



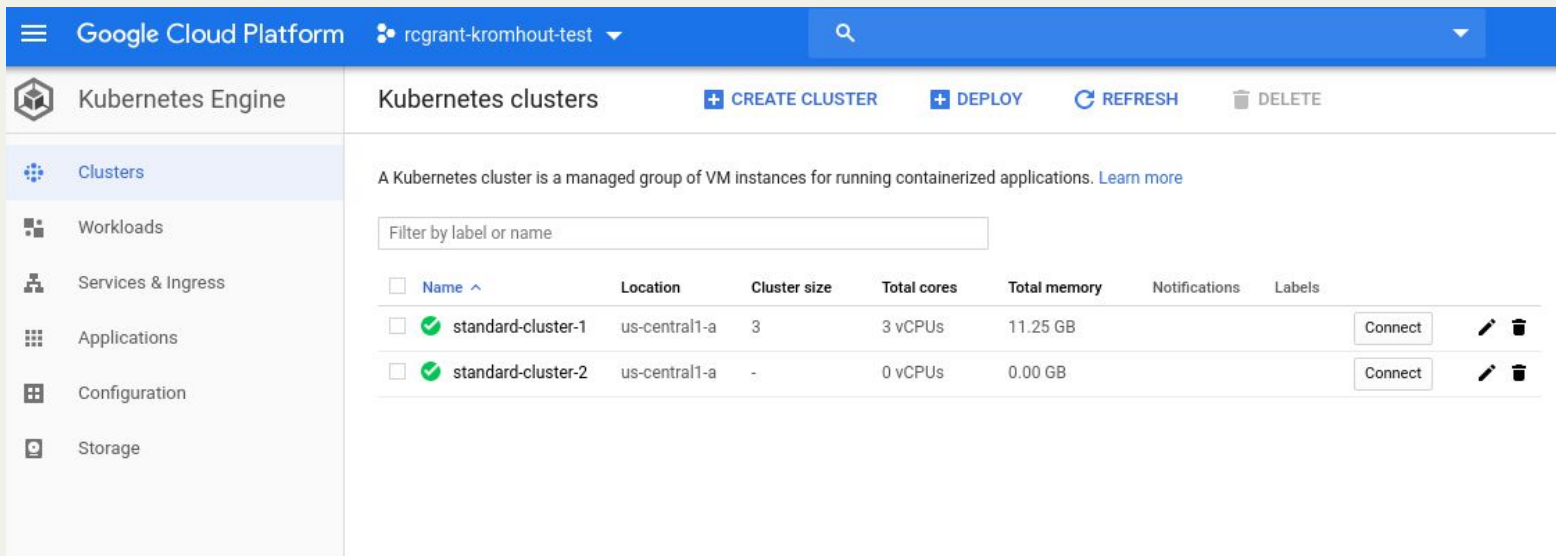
# The “Complex Demo”

- Created by Pavol Mederly of Evolveum, building on the Grouper test docker-compose.
- Includes midPoint, Grouper, Shibboleth IdP, LDAP directory, and sample source of record.
- Builds a nice demonstration and testing environment with several Trusted Access Platform components and sample data



# Kubernetes in GCP

New to GCP and Kubernetes, embarked on a learning exercise.  
Lesson 1: GCP makes building Kubernetes clusters trivial (GKE).  
In about 3 minutes, you have a 3 node cluster with one button access.



The screenshot shows the Google Cloud Platform console interface. At the top, there is a blue header with the Google Cloud Platform logo, the account name 'rcgrant-kromhout-test', and a search bar. Below the header, the left sidebar shows the navigation menu with 'Kubernetes Engine' selected. The main content area is titled 'Kubernetes clusters' and includes buttons for '+ CREATE CLUSTER', '+ DEPLOY', 'REFRESH', and 'DELETE'. A descriptive text states: 'A Kubernetes cluster is a managed group of VM instances for running containerized applications. [Learn more](#)'. Below this is a search filter box labeled 'Filter by label or name'. A table lists two clusters:

<input type="checkbox"/>	Name ^	Location	Cluster size	Total cores	Total memory	Notifications	Labels
<input type="checkbox"/>	<input checked="" type="checkbox"/> standard-cluster-1	us-central1-a	3	3 vCPUs	11.25 GB	<input type="button" value="Connect"/>	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/> standard-cluster-2	us-central1-a	-	0 vCPUs	0.00 GB	<input type="button" value="Connect"/>	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

# Migration to Kubernetes

Lesson 2: Scripts like Kompose are a start, but ...

- No support for converting secrets.
- The complex demo builds needed data into volumes.
- Images need to be available in a repository open to GCP

## SECRETS

Secrets can be easily copied to GCP Kubernetes via kubectl

```
kubectl create secret generic grouper.hibernate.properties --from-file  
configs-and-secrets/grouper/application/grouper.hibernate.properties
```

# Migration to Kubernetes

## VOLUMES

The docker client can be used to dump out volumes into tar files by running temporary containers.

```
docker run --rm --volumes-from complex_grouper_data_1 -v $(pwd):/tmp busybox tar cvf /tmp/complex_grouper_data.tar /var/lib/mysql
```

Kompose takes care of generating YAML files for persistent volume claims to be volumes in kubernetes.

Temporary pods can be used as bridge points to get config files and tar files up into GCP Kubernetes and to untar the files.

# Migration to Kubernetes

## **VOLUMES Continued**

```
kubectl cp configs-and-secrets/grouper/shibboleth/shibboleth2.xml  
grouper-ws-load-data:/etc/shibboleth
```

```
kubectl cp complex_grouper_data.tar grouper-data-load-data:/tmp
```

```
kubectl exec grouper-data-load-data mv /tmp/complex_grouper_data.tar /
```

```
kubectl exec grouper-data-load-data tar xf complex_grouper_data.tar
```

# Migration to Kubernetes

## IMAGES

GCP provides an easy to use image repository with every project. We just need to tag and push the existing local images.

```
docker tag complex_grouper_data gcr.io/rcgrant-kromhout-test/complex_grouper_data
```

```
docker push gcr.io/rcgrant-kromhout-test/complex_grouper_data
```



# Wrapping up

The completed set of YAML, kubectl, and docker commands with a wrapper build script is available on GIT

<https://github.com/ekromhout/midPointComplexDemoKubernetes.git>

Thank you Google for the easy implementation and great documentation.

Also thank you mrbobbytables and recommend this excellent tutorial

<https://github.com/mrbobbytables/k8s-intro-tutorials>

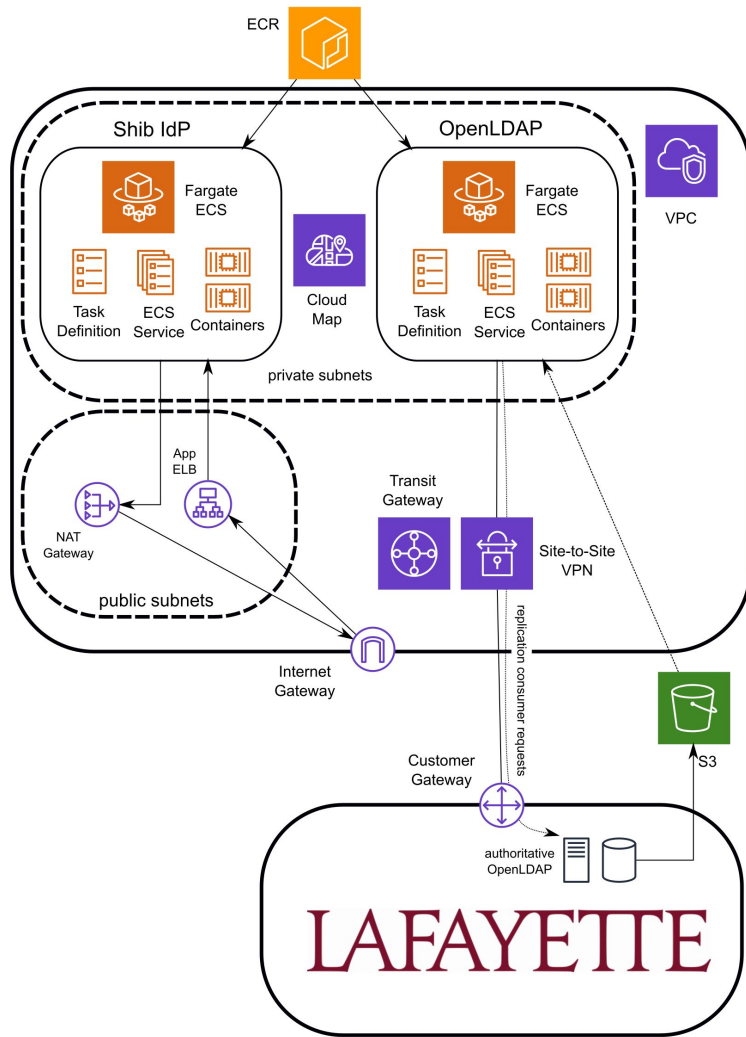
Lastly thank you Sara Jeans for the original title of this presentation:

“Takeout containers for your K8s Lo Mein: Hosting the Trusted Access Platform on Google Cloud Platform Kubernetes GKE”

# LAFAYETTE COLLEGE

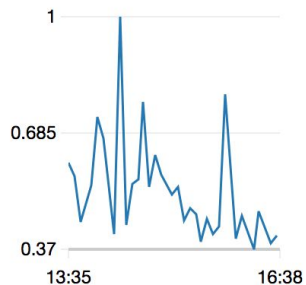
Bill Thompson





### CPUUtilization

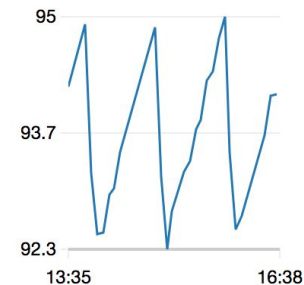
Percent



CPUUtilization

### MemoryUtilization

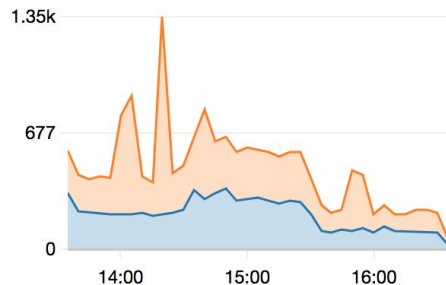
Percent



MemoryUtilization

### RequestCount

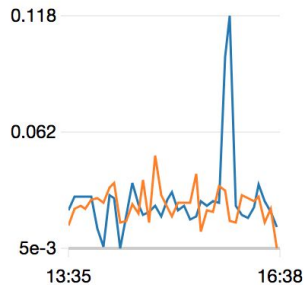
Count



us-east-1d us-east-1c

### TargetResponseTime

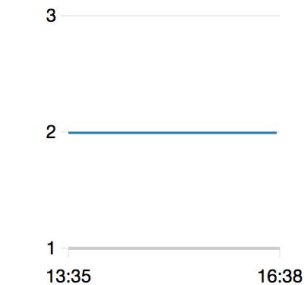
Seconds



us-east-1c us-east-1d

### HealthyHostCount

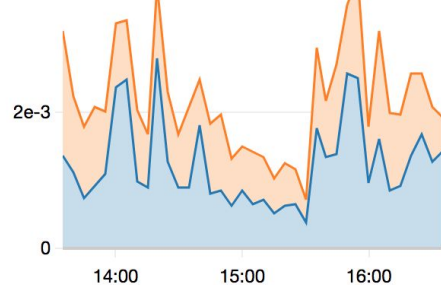
Count



HealthyHostCount

### Average requests per second

Count



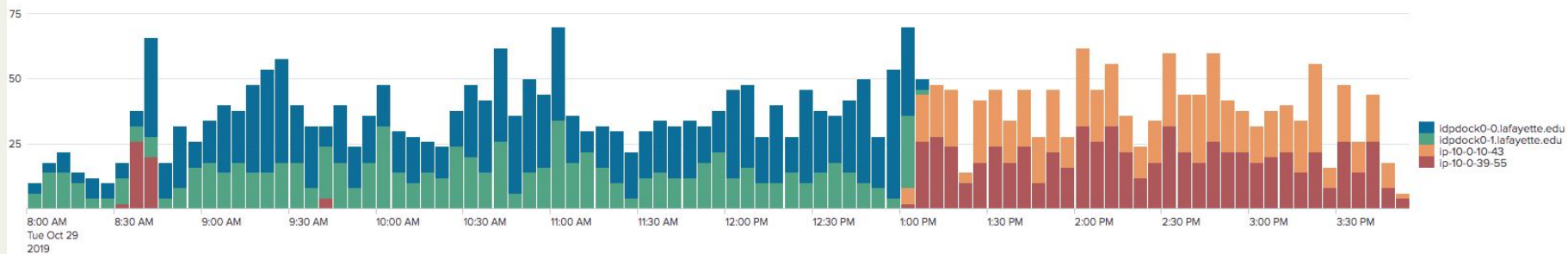
us-east-1c us-east-1d

✓ 3,542 events (10/29/19 8:00:00.000 AM to 10/29/19 3:51:27.000 PM) No Event Sampling ▾

⚠ Job ▾ || ⏏ → 🗑️ ⬇️ ⚡ Fast Mode ▾

Events Patterns Statistics (95) **Visualization**

📊 Column Chart ✎ Format 🗄️ Trellis



# Grouper (and more) Running in AWS

# I

# ILLINOIS

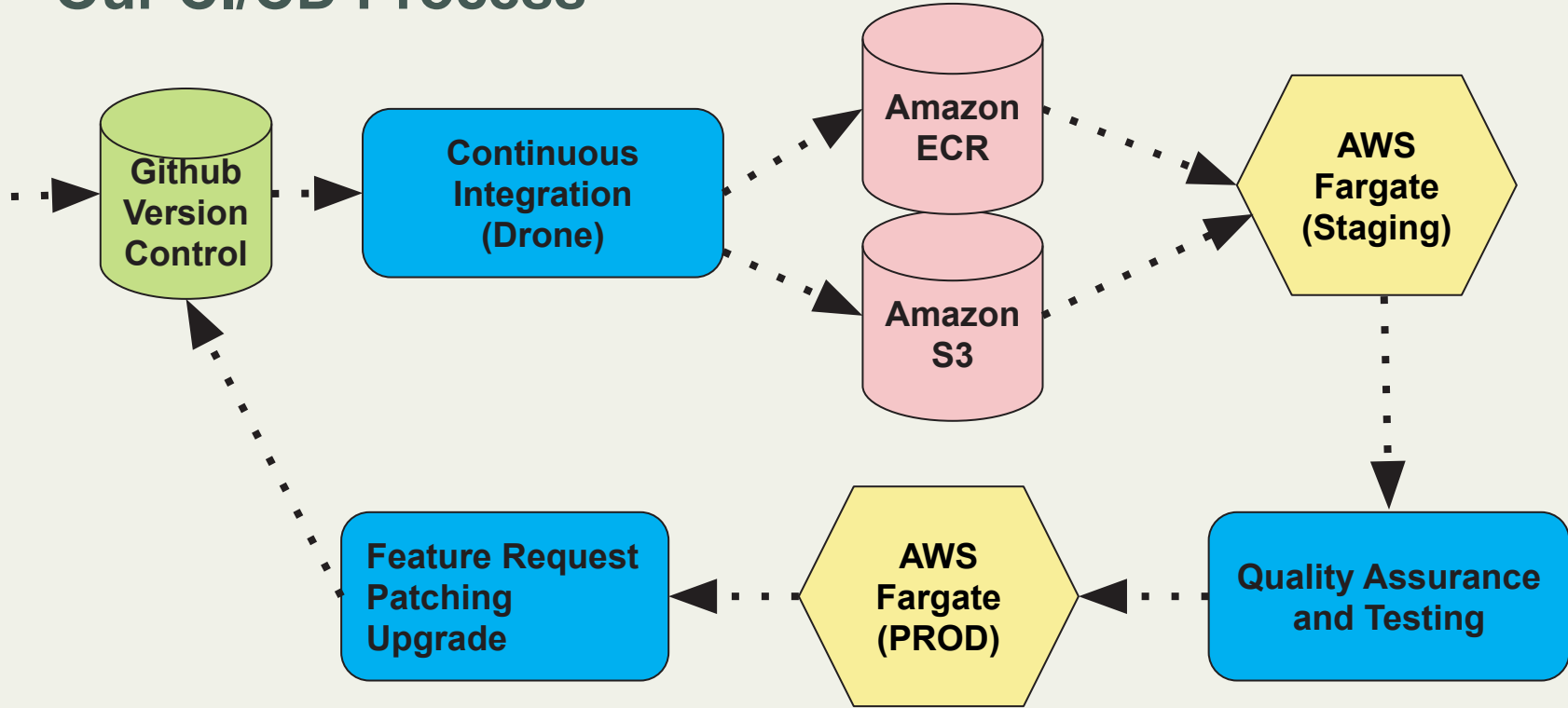
Eric Coleman  
Keith Wessel



# Illinois' Cloud-First Strategy

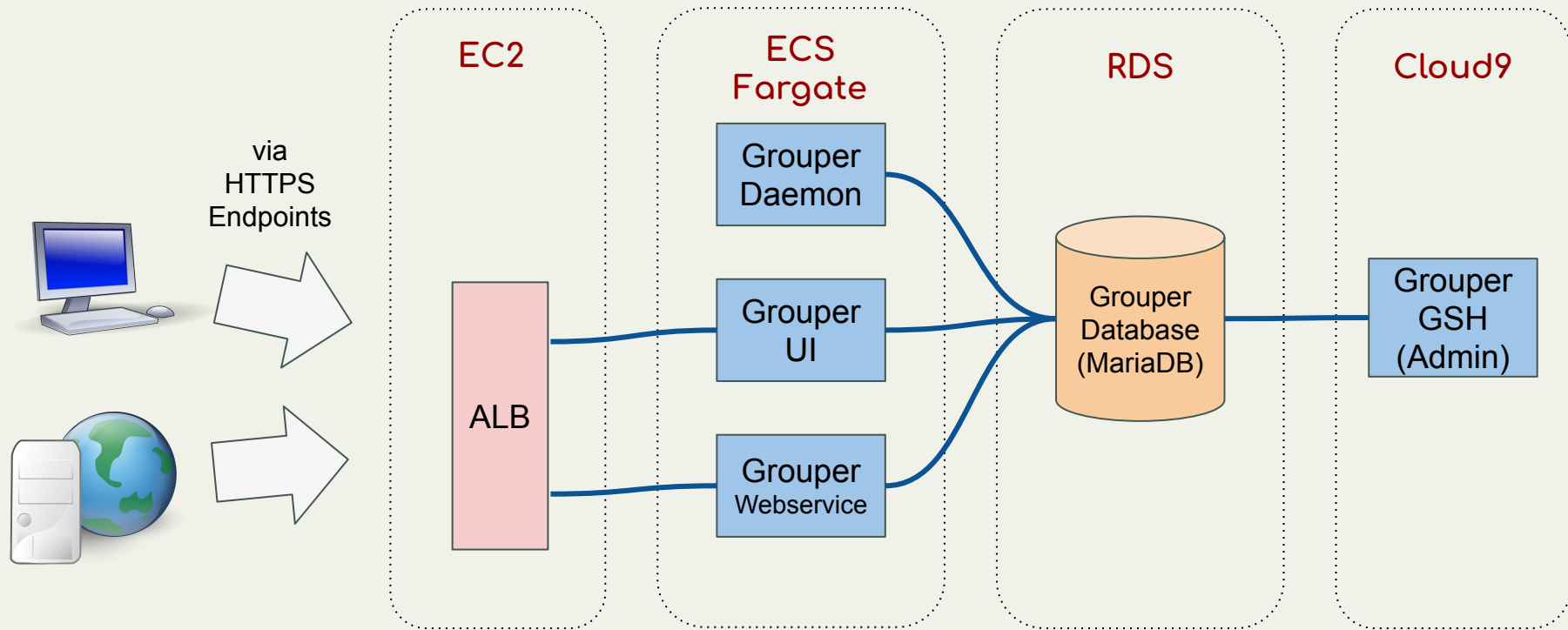
- Organization-wide effort to move to cloud-hosted services
- AWS adopted first, Azure and GCP added later
- Organization-wide "DevOps" model for all central application services
- Preference for AWS EC2 (standalone instances) or AWS ECS Fargate (Docker containers)
- Terraform: cloud infrastructure as code
- Drone: container orchestration
- Github: the repositories

# Our CI/CD Process





# Grouper's AWS Infrastructure at Illinois



## Things We Learned - Secret Storage

- Bad idea: Store passwords in Github
- Good idea: Store passwords in S3
- **Great idea: Use AWS SSM Parameter Store**
- Secrets can also be stored in your CI/CD and built into the image

# Things We Learned - Secret Storage

/service/authman/ldap/bind\_password

Edit Delete

Overview History Tags

Name	Description
/service/authman/ldap/bind_password	Password for ldap/bind_dn
Tier	Type
Standard	SecureString
Last modified date	Last modified user
Thu, 15 Aug 2019 19:56:24 GMT	arn:aws:sts::716155703000:ass role/ApplicationServicesAdmin ois.edu
Value	Version
***** Show	2

containers.json

```
"secrets": [  
  {  
    "name": "SUBJECT_SOURCE_LDAP_PASSWORD",  
    "valueFrom": "/service/authman/ldap/bind_password"  
  },  
]
```

grouper-loader.properties

```
ldap.uofildap.pass.elConfig = ${java.lang.System.getenv().get('SUBJECT_SOURCE_LDAP_PASSWORD')}
```

## Things we learned - Logging

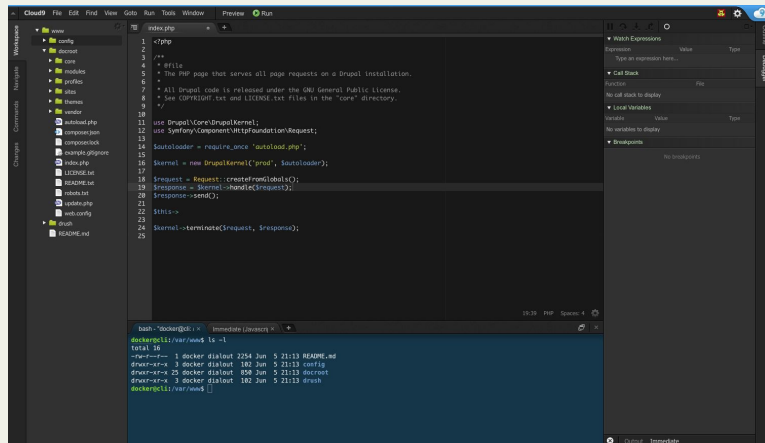
- Ship logs out of container (Cloudwatch)
- Lambda function to pull Cloudwatch into Splunk
- Container-agnostic -- instance handles all Cloudwatch logs
- One HTTP Event Collector per Splunk index



# Things we learned - Admin Console



- SSH into containers is tricky
- Chose AWS Cloud9 IDE
- Inbound access by AWS Role
- Outbound access by Security Groups
- Built-in Linux Shell
  - AWS CLI
  - Docker build
  - Git push
  - Launch Grouper Shell
  - Run MySQL CLI



# Penn Grouper to AWS

Chris Hyzer  
Migrated October 26, 2019

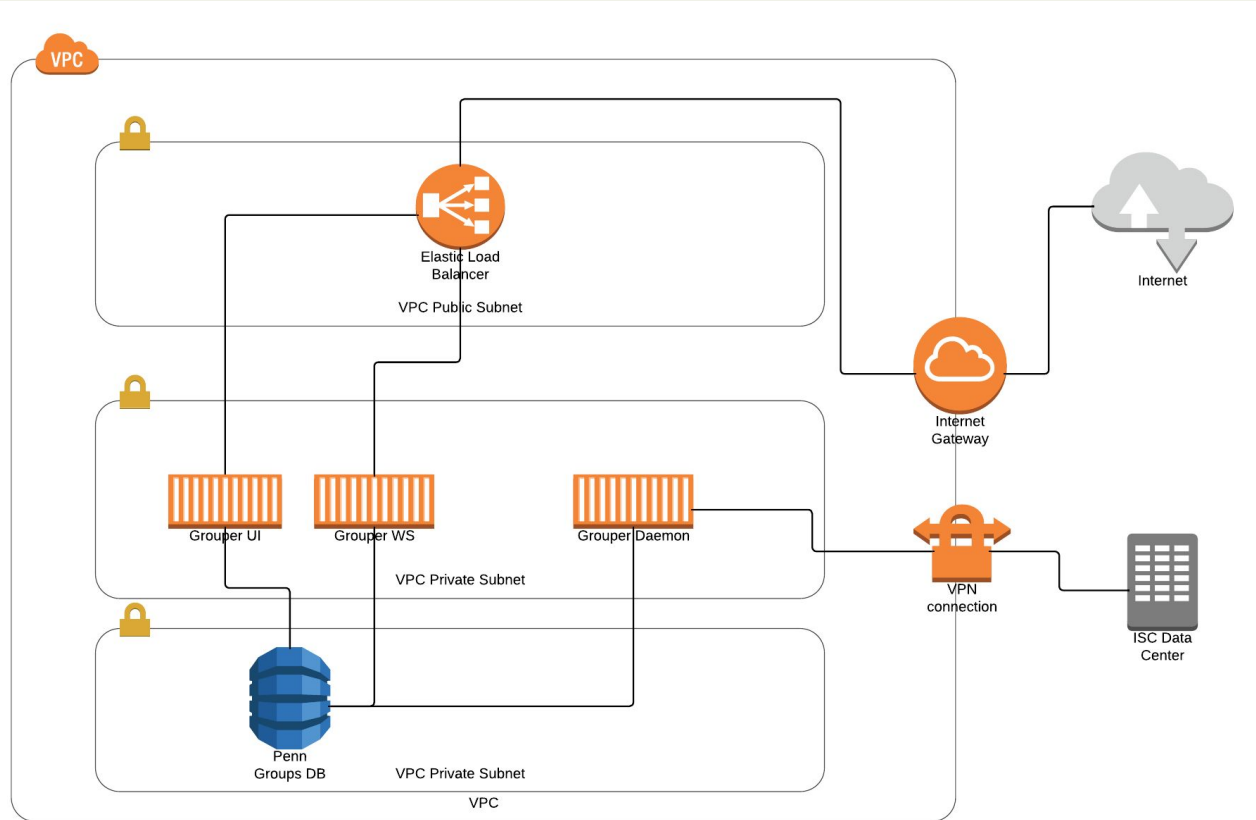


# Legacy architecture

## On prem architecture

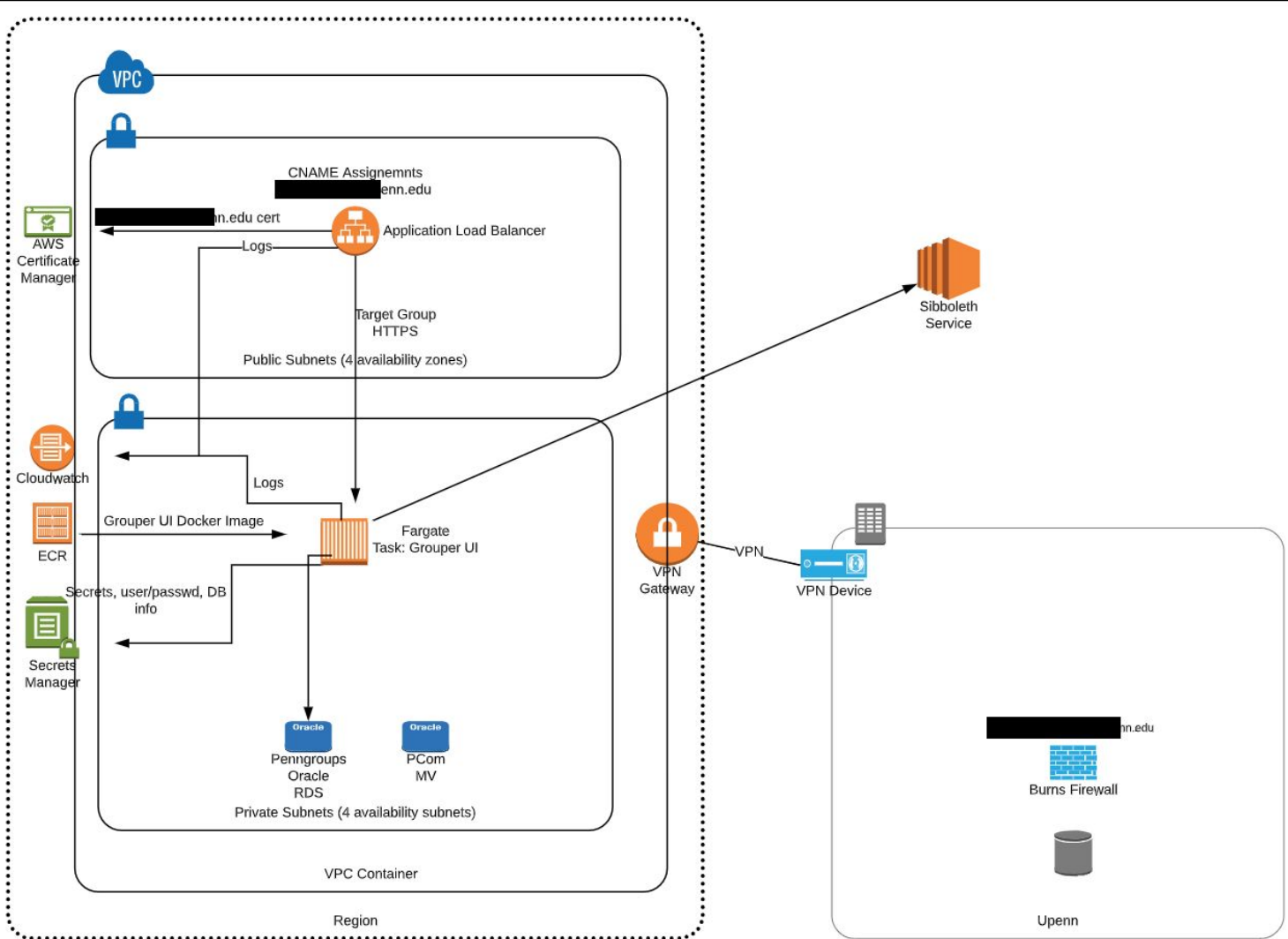
- Oracle shared DB with home grown IDM
- Tomcat not running in container
- Running like other Penn Java webapps
- Need performance improvements (shared Oracle RAC DB)
- Cloud as strategic direction
- Better availability not on prem
- Executive goal
- Did not autoscale
  - 5 daemon
  - 5 UI
  - 5 WS
  - Overkill when non peak

# New design





# New design



# AWS components

- Gitlab - Private repo for each env. Webhooks for auto-deploy
- Jenkins - Deployment automation. Gitlab tools for integrated automations.
- Slack - Output Jenkins job logs to channel
- AppELB - End user application endpoint. HTTPS end to end
- ECR - Docker container registry
- Fargate - Application container hosting
- Secrets Manager - Store db and morph passwords
- Cloudwatch - Send all AWS service related and application logs to log groups
- RDS for Postgres- Encrypted data at transit and rest. Multi AZ for prod.
- Route 53 - Create cname entry for RDS endpoints.

# Envs

- Prod
  - UI: min 1 container
  - WS: min 3 containers
  - Daemon: 2 containers (8 gig memory)
  - GSH: 1 container
- Non-prod
  - UI: min 1 container
  - WS: min 1 container
  - Daemon: 1 container
  - GSH: 1 container
- Started with 3 envs, we could spin up another if needed

## Configuration in DB

- If configuration in DB that is migrated with DB migration
  - Compare config files and import into UI
- Need to make sure firewalls are open to / from all endpoints
- Most passwords encrypted in database
  - Except DB and morph
  - Passwords from password manager in env variables

# Data migrations

- Migrated from Oracle to postgres
- Needed some Grouper database back on site
- Needed subject source kept in sync from on-site to AWS
- Need Grouper memberships for shib copied to shib database
- Generally using “[Grouper SQL database provisioning](#)”
  - Need to use this more and make it incremental as well as full sync

# Performance

- Our Oracle on-prem had performance issues
- Aurora postgres is peppier
- Latencies can be a problem (e.g. provisioning to LDAP)
- WS are faster though extra latency

## WS migration experience

- Smooth
- Some performance issues that were resolved
- Timeout of large queries had to be adjusted in some places (e.g. ELB)
- Did not have memory set correctly

## UI migration experience

- Smooth
- Did not have memory set correctly
- Missing some files that were on server and not in container overlay
- Some links to old URLs did not migrate correctly
  - Bookmarks



## Daemon migration experience

- Bumpy
- Did not have memory set correctly
- Needed a lot more memory (went from 5 servers down to 2)
- Data migrations needed to be dealt with
- Change “grouper” database connection to point to old database
- PSPNG slower in cloud (due to latency?)
  - We also coincided with more usage
- Wackamole of issues

## Post migration

- Made some fixes
- Like the git -> deploy model
  - Approval request sent via slack
- Some changes made quickly and easily
- Going well

# Internet2 Collaboration Platform

Chris Hubing  
Running in AWS



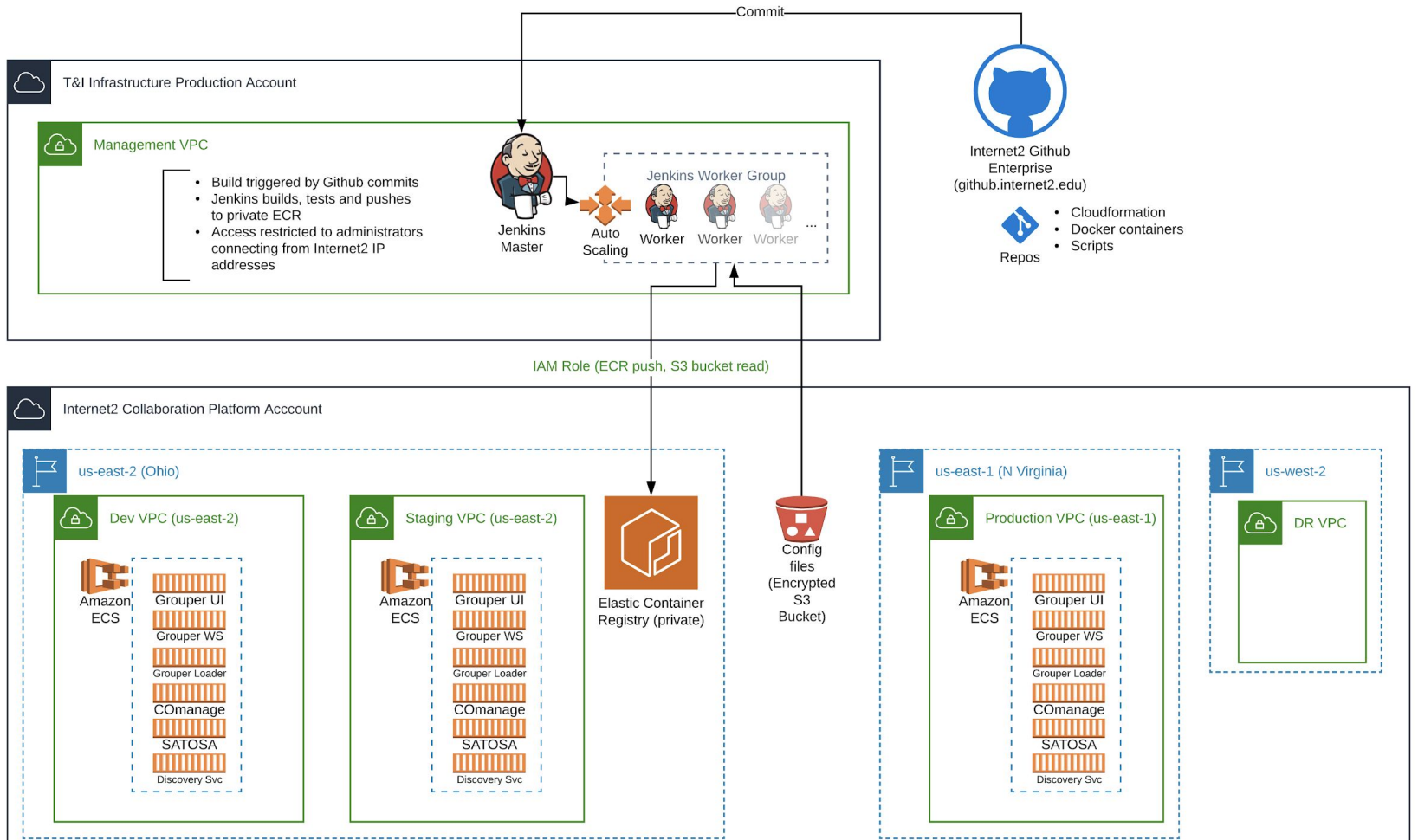
# Internet2 Collaboration Platform

- Using InCommon Trusted Access Platform containers
  - COmanage (container) as registry
  - Grouper (containers UI, WS, Loader) for Access Management
  - SATOSA (container) as an IDP proxy
  - RabbitMQ (container) as message queue
  - Midpoint (container) for provisioning (not in prod yet)

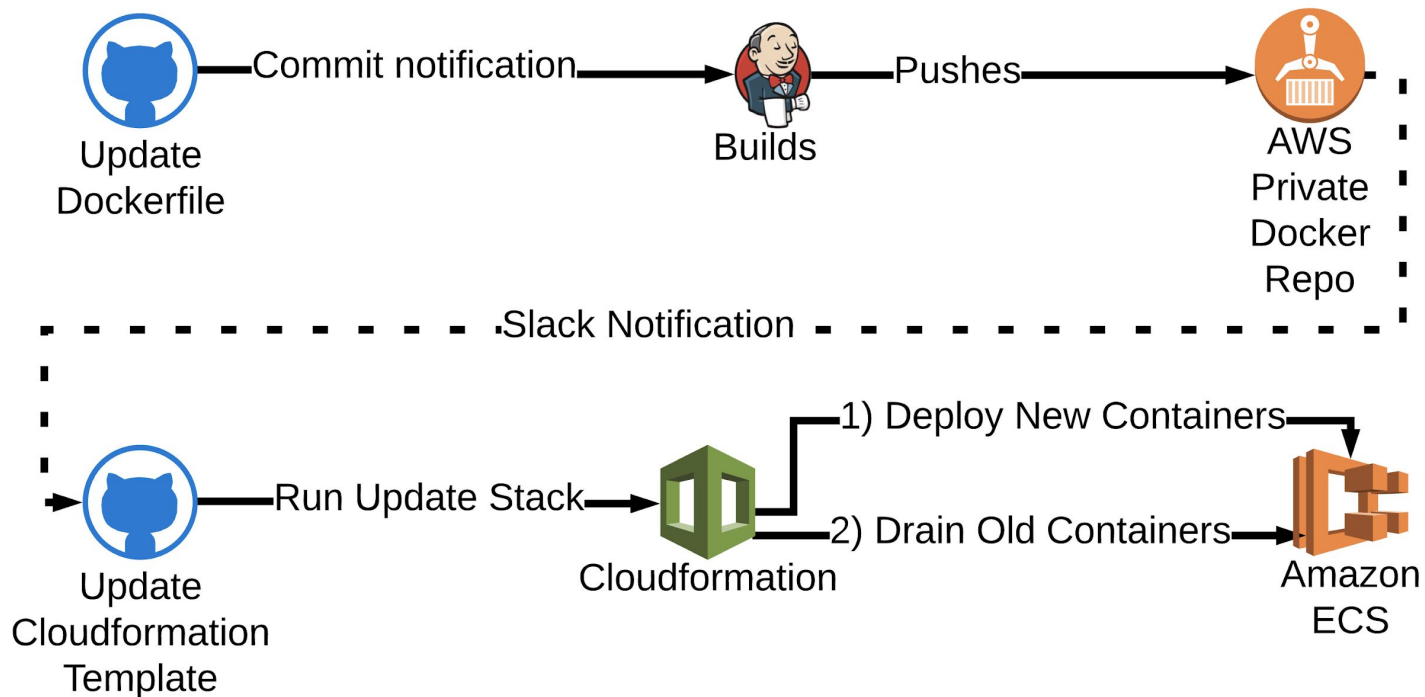
# Internet2 Collaboration Platform

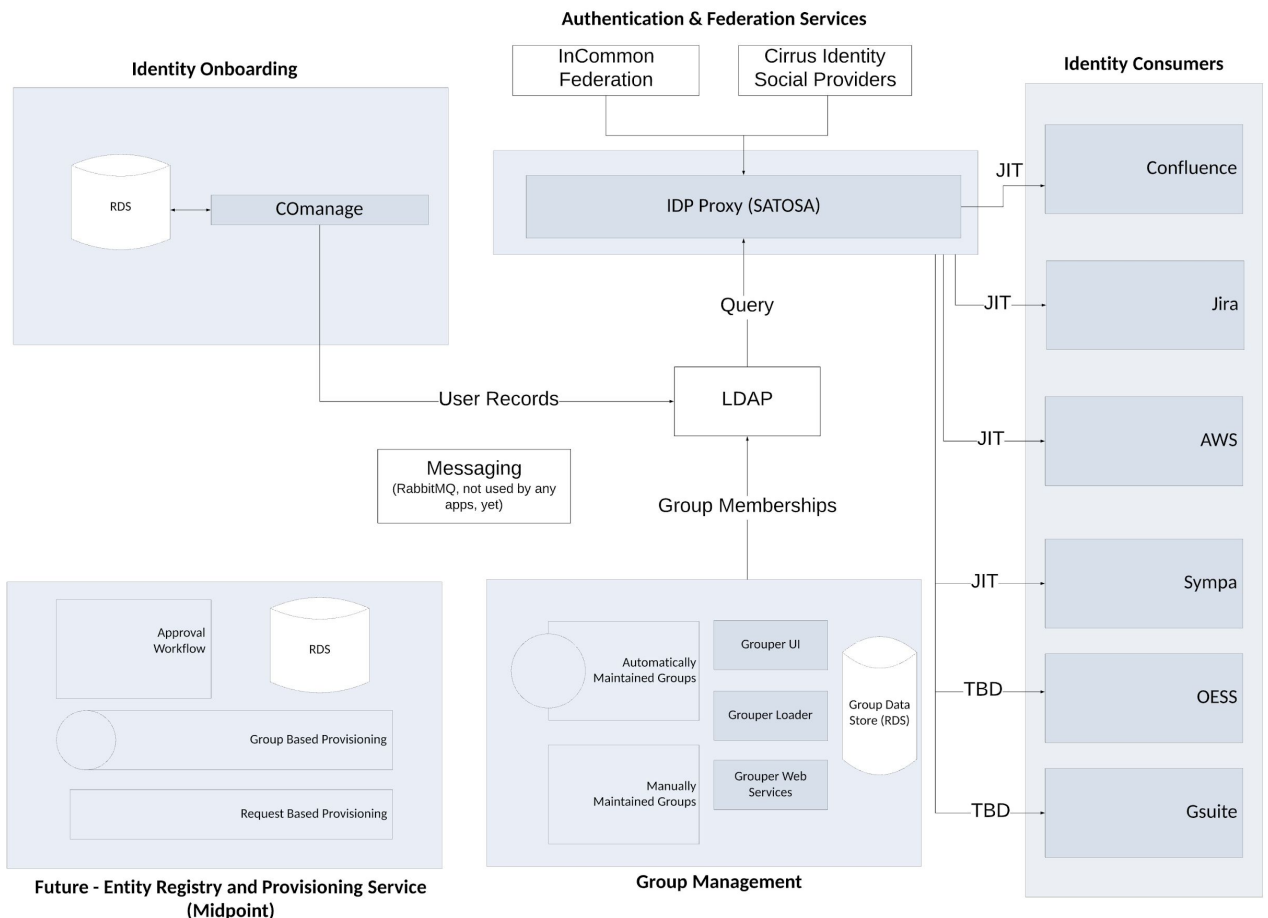
- Confluence, JIRA, Sympa, AWS, *GitHub*, *Jenkins* are SAML domesticated
- Running in AWS Elastic Container Service (ECS)
- Github Enterprise for repositories ([github.internet2.edu](https://github.internet2.edu))
- Jenkins automated builds
- Cloudformation (JSON template) for infrastructure as code

# Internet2 Collaboration Management Platform



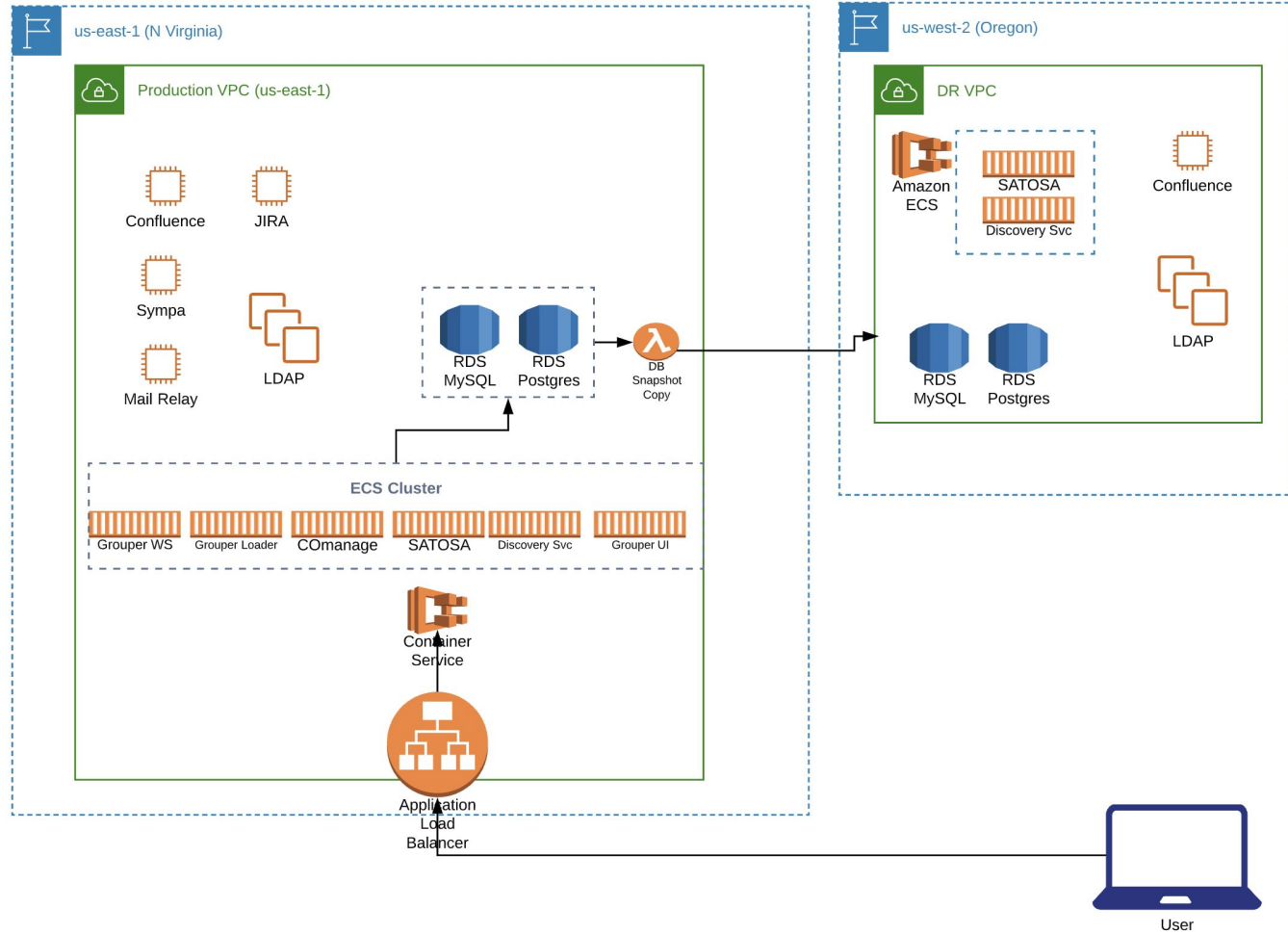
# Container Update Process - Midday Deployment







# ICP - Production and DR



# Cloudformation Resources Utilized

12 "AWS::EFS::MountTarget"	2 "AWS::Logs::LogGroup"
9 "AWS::Route53::RecordSet"	2 "AWS::ElasticLoadBalancingV2::ListenerRule"
7 "AWS::ECS::TaskDefinition"	2 "AWS::EFS::FileSystem"
7 "AWS::ECS::Service"	1 "AWS::S3::BucketPolicy"
6 "AWS::ElasticLoadBalancingV2::TargetGroup"	1 "AWS::S3::Bucket"
4 "AWS::ElasticLoadBalancingV2::LoadBalancer"	1 "AWS::ECS::Cluster"
4 "AWS::ElasticLoadBalancingV2::Listener"	1 "AWS::EC2::SecurityGroupIngress"
4 "AWS::EC2::SecurityGroup"	1 "AWS::CloudTrail::Trail"
2 "AWS::RDS::DBSubnetGroup"	1 "AWS::AutoScaling::LaunchConfiguration"
2 "AWS::RDS::DBInstance"	1 "AWS::AutoScaling::AutoScalingGroup"

# Internet2 Collaboration Platform - Lessons and Future

- Have a VM in the VPC you can spin up containers for debugging/testing
- Velocity and Quality of deployments has increased
- Secrets Manager - currently secrets are stored in encrypted S3 bucket (but be careful switching RDS to it)
- Fargate EKS (just announced at AWS re:Invent last week)
- Move Github and Jenkins into Prod
  
- **Go grab our/your code ([github.internet2.edu](https://github.com/internet2edu))**

## Questions?

**Keith Wessel** University of Illinois - Urbana-Champaign

**Ethan Kromhout** University of North Carolina - Chapel Hill

**Erik Coleman** University of Illinois - Urbana-Champaign

**William Thompson** Lafayette College

**Chris Hyzer** University of Pennsylvania

**Christopher Hubing** Internet2