

UC Office of the President

Sessions

Title

Our Journeys to DevOps across UC Library Development Teams

Permalink

<https://escholarship.org/uc/item/0hv8h2sg>

Authors

Brady, Terry
Critchlow, Matt
Schwab, Ryan
et al.

Publication Date

2025-10-23

Supplemental Material

<https://escholarship.org/uc/item/0hv8h2sg#supplemental>

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Our Journeys to DevOps across UC Library Development Teams

Panelists

- Terry Brady, CDL
 - <https://github.com/terrywbrady>
- Matt Critchlow, UCSD
- Ryan Schwab, UCSC
- Alexandra Dunn, UCSB
- Dan Schmidt, UCB

Overview

- Team Presentations
 - 5 minute overview of each team's DevOps journey
- Panel Questions
- Audience Questions

UCB

DevOps at Berkeley — Org

- Head of Library IT
 - Head of DevOps
 - 1x Windows Admin
 - 1x Linux/DC Admin
 - 1x DevOps Specialist
 - Head of Applications
 - 5x developers
- Very wide portfolio (~60 services)
- Support for novelty preferred over robustness (but this is changing)

DevOps at Berkeley — History

- Started at zero
 - No version control
 - No packaging
 - Code/language sprawl
 - Everything opaque / siloed
 - ~150 virtual machines
- DevOps role created to fill knowledge gap
- Familiar action plan
 - Transparency and openness
 - GitOps (Ansible, Terraform)
 - Containerization — The North Star
 - Standardization

DevOps at Berkeley — Constraints

- Digitization workflows
 - Multiple teams in critical path
 - 300TB storage
 - Manual QA
 - Requires developer / DevOps time
- Budget (esp. storage)
- Skills
 - Wide/changing portfolio makes it harder to optimize

DevOps at Berkeley — Currently

- On-prem (vSphere)
 - Standard IAC stack: Ansible, Packer, Terraform
 - 28 total VMs (big chunk inherited from outside LIT)
- Containerized
 - Docker Swarm (dev, staging, and prod clusters)
 - Linux side fully containerized (minus Dataverse)
 - NFS volumes; Traefik ingress/routing
- Code in GitHub (with some remaining in on-prem GitLab)
 - GitHub Actions builds, tests, tags, etc. images
 - On-prem declarative Jenkins for sensitive infrastructure
 - Ansible drives deployments (stacks tracking specific image tags)

DevOps at Berkeley — Dev to Prod

- Dev starts work locally
 - `git pull && docker compose up -- build`
- Push to feature branch —> tagged `:feature` —> deploys preview
 - Automated build / test
 - Metadata includes SBOM, attestations, provenance, etc.
 - Build also applies other tags (inc. build-specific)
- Merge to main —> tagged `:latest` —> deploys staging
 - More testing, inc. manual QA w/stakeholders
- Create a release (semver) —> tagged `:<versions>` —> deploys production
 - Can be more deliberate as needed (e.g. infra coordination)

DevOps at Berkeley — Pain/Growth Points

- Swarm
 - Batteries included and simple, consistent CLI, but...
 - ... very limited ecosystem
 - ... poor extension API (e.g. container lifecycle)
- Mix of GitHub/GitLab/Jenkins
 - Consolidate to GitHub / Actions
 - Requires new platforms
 - Accelerate shift to pull-based convergence model
- Observability
 - Zabbix drowns in orchestration data

UCSB

DevOps at UCSB – how things have been done

- (Long ago) Ansible, Jenkins, on-premises VMs
- We adopted Kubernetes and Helm over 5 years ago
 - EKS (benefits and drawbacks)
- GitLab CI/CD
 - Significant manual configuration but extremely flexible

DevOps at UCSB – where things are moving

- Flux for static cluster objects
- The Bitnami Affair and the dangers of centralization

UCSC

DevOps at UCSC - Laying the foundations

Git

- Version control is a prerequisite for DevOps
- All application code is in version control
- Server configuration is now in version control
- Experimenting with using git for metadata spreadsheets
- Ideal state: Everything is version controlled

DevOps at UCSC - Laying the foundations

Docker + Kubernetes

- Our local dev environments run in containers
- A small number of our apps run in containers in production
- We will soon implement the Surfliner project using Kubernetes
- More applications will likely be migrated to use these workflows

DevOps at UCSC - Laying the foundations

Jenkins

- Deployment automation tool
- We used this for our previous implementation of Hyrax (Digital Asset Management System)
- Will evaluate this alongside other options such as Github actions and AWS CodePipeline

DevOps at UCSC - Laying the foundations

Infrastructure as Code: Cloudformation + Chef

- Cloudformation builds infrastructure in AWS using JSON/YAML templates
- Chef automates the provisioning of servers (Ansible does this too)
- All server and network infrastructure is now described in version-controlled code
- All system builds including package installation and configuration are now automated

DevOps at UCSC - What's Next

Process Changes

- Commit early and often (continuous integration)
- Write a lot more tests, and run the test suite on every push
- Deploy on every successful run of the test suite
- Repeat this process often and in small increments (continuous deployment)
- Developers and SysAdmins working closely together

UCSD

DevOps at UCSD - How Things Have Been Done

- 1 development team
 - Multiple applications supported by single team
 - Deployments via SSH to dedicated VMs
 - Deployments manually triggered by Gitlab ChatOps
- 1 systems administration team
 - Manual provisioning of VMs for applications
 - System Administrators administer VMs
 - Developers are given SSH access to VMs as needed
 - Manage external Dependencies like Solr, PostgreSQL

DevOps at UCSD - How Things Are Evolving

- 1 library DevOps team
 - Multiple applications supported by single team
 - Applications are all containerized
 - Applications are packaged and deployed as a Helm Chart
 - Deployments occur via Gitlab Pipelines
 - Manage 2 primary Kubernetes clusters (review and production)
 - Shared management of external services via Git(lab) repositories

Example: Library Digital Collections (Today)

- 4 applications
 - Mix of Java and Ruby
 - Different deployment patterns, all to Ubuntu VMs
- **No observability just logs**
- **Dependent services not managed via Git(lab)**

Example: Library Digital Collections (Future)

- 7+ applications via Project Surfliner (with UCSB)
 - Primarily Ruby backend, JavaScript frontend
 - Standardized deployment pattern with Helm Charts using Container Images
- **Observability via OpenTelemetry / Honeycomb**
- **Monitoring/Metrics via Prometheus / Grafana**
- **Dependent services managed via Git(lab)**

Example: Library Digital Collections (Process)

- Collaboration
 - Building shared solutions with UC Santa Barbara
 - Ensure Infrastructure as Code
 - Ensure Cloud Native deployment supports on-prem and vendor solutions
- Learning
 - Prefer shared learning to speed. Zoom and Slack Huddles
 - Very supportive Manager and Director
 - UCSB/Surfliner collaboration helping us support each other during challenging times

CDL

DevOps at CDL - How Things Have Been Done

- 4 programs
- 3 development teams
 - Multiple systems supported by each team
- 1 central systems administration team
 - Centralized AWS Account
 - System Administrators have AWS Console Access
 - System Administrators administer resources
 - AWS Permissions granted to servers
 - Developers are given access to servers

DevOps at CDL - How Things Are Evolving

- 4 programs
- 3 development teams
 - Multiple systems supported by each team
 - Each team administers a separate AWS Account
 - Resources are created with “Infrastructure as Code”
 - Long living resources such as servers are discouraged
 - Resources are regenerated from code rather than patched
- 1 central systems administration team
 - Set guardrails for the separate AWS accounts (networking, security, cost)
 - Consult with teams and provide tooling

Example: Merritt Preservation System (Today)

- 1,700 TB of cloud storage
- 1 Database
- 7 services
 - Incrementally enhanced for container deployment
- **27 servers**
- **5 TB file system for processing content**

Example: Merritt Preservation System (Future)

- 1,700 TB of cloud storage
- 1 Database
- 7 services
- **Dynamically Scalable Number of Containers**
- **Utilize Cloud Storage for Processing**

Example: Merritt Preservation System (Process)

- As subservices migrate to our new account
 - Ensure Infrastructure as Code
 - Ensure *Dispose and Recreate* of Resources
 - Leveraging Cloud Storage for content processing
 - Discover ways to dynamically scale

Panel Questions

What are your biggest challenges with
DevOps?

How is your team learning DevOps?

Audience Questions

Additional Panel Questions (time permitting)

- What aspects of DevOps excite you the most?
- How are the roles of a developer and a system administrator evolving on your teams?