

Developing for the Cloud, in the Cloud

Michael B. Klein (he/him/his)
Software Development Tech Lead
Repository & Digital Curation
Northwestern University Libraries

Internet2 Technology Exchange 2022
December 8, 2022 / Denver, Colorado

About Me



Photo: Joshsukoff - Own work, CC BY-SA 4.0
<https://commons.wikimedia.org/w/index.php?curid=124363357>

About the Team



Michael Klein
Tech Lead/Developer
@mbklein



Karen Shaw
Senior Developer/Scrum Lead
@kdid



Brendan Quinn
Senior Developer
@bmquinn



Adam Arling
Senior Developer
@adamjarling



Mat Jordan
Developer
@mathewjordan



David Schober
Team Lead/Product Owner
@davidschober



Veronica Robinson
Service Lead
@vlsrobinson

Our (Current) Projects

Name	Description	Platform	Audience	Deployed Via
Arch	Institutional Repository	Ruby on Rails	Public	Docker
AVR	Audiovisual Repository	Ruby on Rails	Faculty/Students	Docker
Meadow	Repository Asset Management	Elixir/Phoenix React	Library Staff	Docker
Digital Collections	Digital Collections Discovery & Access Website	Next.js/React	Public	Hosted Single-Page App
DC API	Digital Collections Discovery & Access API	NodeJS	Public	AWS Serverless
IIIF	Image Server	NodeJS	Public	AWS Serverless

Partnerships & Community Involvement



Samvera Community

- Open source repository framework
- 30 partners / dozens of users
- Underpinnings of both AVR & Arch
- Full projects & components
- Committers, contributors, and product owners from our team

IIF Consortium

- International Image Interoperability Framework
- Open standards for delivery of digital objects at scale
- NUL's IIF projects:
 - Serverless IIF service
 - IIF-compatible front-end components

Application Infrastructure

Meadow

1 PostgreSQL Database
5 OpenSearch Indexes
11 Lambda Functions
6 S3 Buckets
13 SQS Queues
1 MediaConvert Pipeline
1 A/V Streaming Server
1 IIIF Service
1 LDAP Server
S3 Object Triggers
EventBridge Rules

AVR

1 PostgreSQL Database
1 Fedora Repository
1 Zookeeper Config
1 Solr Index
4 S3 Buckets
18 SQS Queues
1 Redis Cache
1 MediaConvert Pipeline
1 A/V Streaming Server

Arch

1 PostgreSQL Database
1 Fedora Repository
1 Zookeeper Config
1 Solr Index
3 S3 Buckets
3 SQS Queues
1 Redis Cache
S3 Object Triggers

Guiding Principles

- Configuration over customization
- Let developers be developers
- Ease of onboarding
- Stop racing against the depreciation of our laptops

Our Toolchain

Elixir 1.14 /
OTP 25.1

NodeJS 16.x
and 14.x

Ruby 2.7

Python 2.7
and 3.10

Git / GitHub

npm • yarn
bundler
hex • poetry

Terraform

Docker
Engine

AWS CLI

AWS SAM
CLI

AWS ADFS
Login

MkDocs

Terraform

- Infrastructure as code
- Configuration: Human readable, declarative resource definitions
- Plan: What needs to be created/changed/destroyed
- Apply: Make the changes dictated by the plan
- State: A record of how things were the last time we checked
- Workspace: A single named instance of state data

Iteration 1: docker-compose

Goal: make local development possible

- Individual `docker-compose.yml` for every project
- Official images:
 - PostgreSQL, Redis, Elasticsearch/OpenSearch, Solrcloud
- Third party images:
 - minio (S3), go-aws (SQS)
- Bespoke/customized images:
 - Fedora Repository, IIF, Media Streaming, LDAP

Iteration 2: devstack v1

Goal: make (most) developers' lives easier

- Custom wrapper for docker compose (~250 LOC)
- Shared `docker-compose.yml`
- Handled complicated orchestration of 13 different services
- Independent dev/test environments
- Data persistence, but easy to tear down/start from scratch

Iteration 3: devstack v2

Goal: more AWS service emulation

- Replaced minio & go-aws with [localstack](#)
- Configuration now includes **custom Terraform manifests**
- Lambda functions
- HTTP and REST APIs via API Gateway
- Looking ahead: Step Functions, EventBridge Rules

Unresolved Issues


- Complex runtime requirements
- Increasingly difficult to replicate/emulate on a workstation or laptop
- Architecture decisions constrained by development resources
- Need to develop and test with larger fixtures and data



Iteration 4: devstack as a service


Goal: compatibility & convenience

- Hosted on an AWS EC2 instance
- Uses a combination of shared and individual resources
- Uses actual cloud services instead of mocks and emulators
- Gets beyond laptop resource limits
- Persistent at the developer level
- Supports individual user preferences
- Easy to maintain
- Simple to replace




Developer EC2 Instance


```
DEV_PREFIX=bob  
DEV_ENV=dev
```



bob-dev-ingest
bob-dev-preservation
...



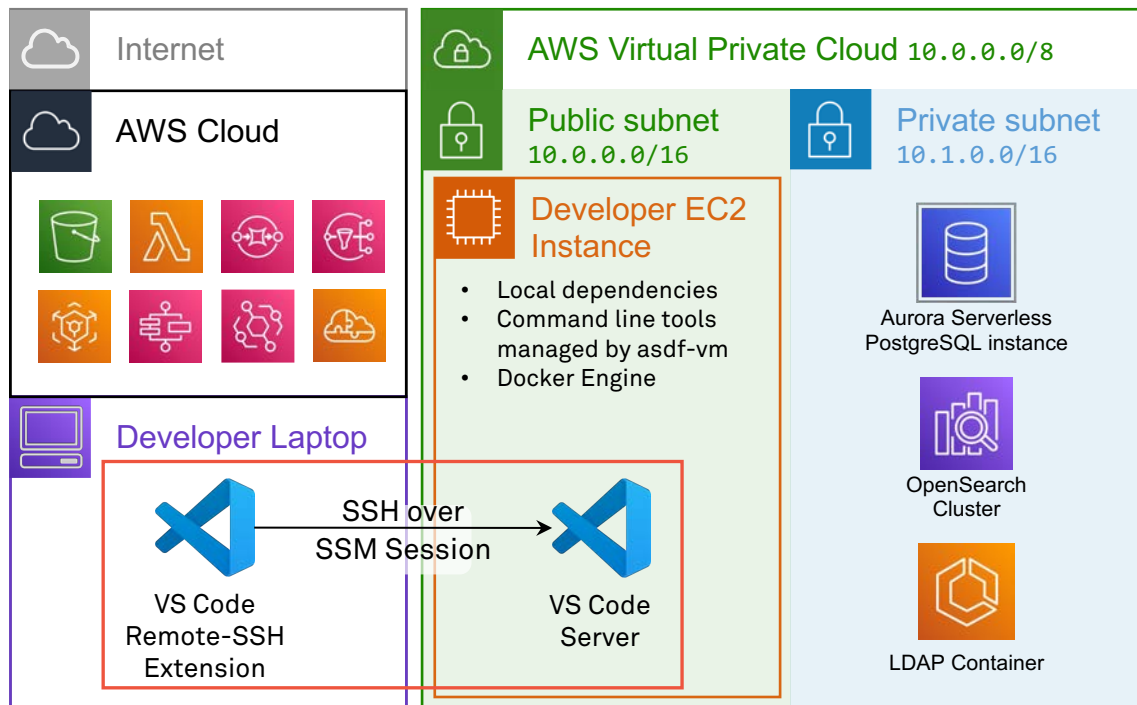
bob-dev-ingest-file-set
bob-dev-initialize-dispatch
bob-dev-extract-mime-type
bob-dev-copy-file-to-preservation
...

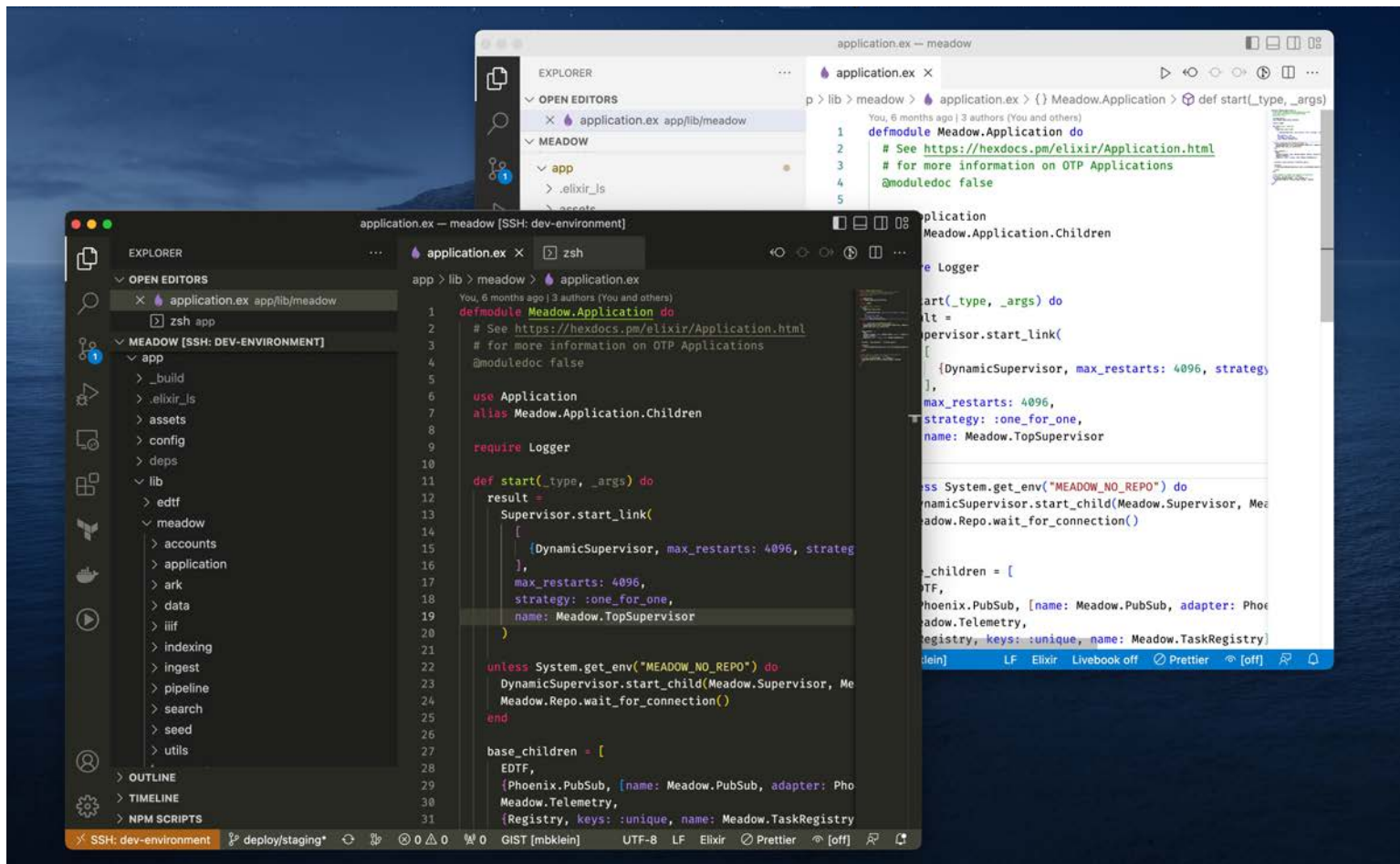


bob-dev-meadow

Technical Details

- Shared Resources
- Individual Resources
- Configuration
- User Scripts & Utilities
- Automated Setup, Teardown & Replacement





One-Time Setup

```
$ cd common
```

```
$ terraform init
```

```
$ terraform apply
```

Total build time: ~15 minutes

- Virtual Private Cloud
- Database cluster
- OpenSearch cluster
- DNS zone
- IIF server
- Shared Lambda functions
- LDAP server
- Shared access policies, roles, and environment configuration

New Developer Setup

```
$ cd individual
```

```
$ terraform init
```

```
$ terraform workspace select bob
```

```
$ terraform apply
```

- Developer VM
- S3 Buckets x2
- SQS Queues x2
- Policies & roles
- Initial system config

Total build time: ~5 minutes

Time until init script completes:
~20 minutes

Maintenance & Support

AWS Systems Manager

Quick Setup

Search:

Name	Owner	Platform types
AWS-RunShellScript	Amazon	Linux, MacOS

Description
Run a shell script or specify the commands to run.

Document version
Choose the document version you want to run.
1 (Default)

Command parameters

Commands
Required: Specify a shell script or a command to run.

```
1 sudo -H su ec2-user sh -c \  
2 "rsync -rtags -e ssh golang $& \  
3 esaf install golang_1.19.2 $& \  
4 esaf global golang_1.19.2 \  
5 "
```

Working Directory
Optional: The path to the working directory on your instances.

Execution Timeout
Optional: The time in seconds for a command to complete before it is considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48 hours).
3600

Target selection

Choose a method for selecting targets.

Specify instance tags
Specify one or more tag key-value pairs to select instances that share those tags.

Choose instances manually
Manually select the instances you want to register as targets.

Choose a resource group
Choose a resource group that includes the resources you want to target.

Resource group
Select the resource group that you want to use as a target. [View resource groups](#)
dev-environment-resources

AWS Systems Manager

Quick Setup

AWS Systems Manager > Maintenance Windows > Window ID: mw-04ace7c876ddc1d39 > Description

Window ID: mw-04ace7c876ddc1d39

Operations Management

Explorer
OpsCenter
CloudWatch Dashboard
Incident Manager

Application Management

Application Manager
AppConfig
Parameter Store

Change Management

Change Manager
Automation
Change Calendar
Maintenance Windows

Node Management

Fleet Manager
Compliance
Inventory
Hybrid Activations
Session Manager
Run Command
State Manager
Patch Manager
Distributor

Shared Resources

Documents

Description | Tasks | History | Targets | Tags

Window ID: mw-04ace7c876ddc1d39

Name: dev-environment-backup

OpsCenter: dev-environment-backup

Description: Run home directory backup on all developer environment instances

State: Enabled

Duration: 1 hour

Crn/Rate expression: cron(D 23 * * Fri *)

Next execution time: Sat, Dec 3, 2022 at 5:00:00 AM UTC

Window schedule timezone: America/Chicago

Window schedule offset: -

Allow unregistered targets: Yes

Cutoff point: 0 hours before window closes

Window start date: -

Window end date: -

Tasks

Window task ID	Priority	Name	Task ARN	Type	Targets
<input type="radio"/> 01c4c7b8-4255-4d92-9b43-ce05a797950	20	-	AWS-RunShellScript	RUN_COMMAND	1
<input type="radio"/> b3c679a6-4250-4002-8555-ab51a36564ae	10	-	AWS-StartC2Instance	AUTOMATION	1

Targets

Window target ID	Name	Targets	Owner information
<input type="radio"/> 3c0760aa-9936-4128-9853-425aae2f213e	dev-environment-backup-targets	2	-

Maintenance & Support

The screenshot shows the AWS Management Console interface for connecting to an EC2 instance. The breadcrumb navigation at the top reads: EC2 > Instances > i-0731396381ba172c6 > Connect to instance. The main heading is "Connect to instance" with an "Info" link. Below the heading, a message states: "Connect to your instance i-0731396381ba172c6 (nmbk-dev-environment-ide) using any of these options". There are four tabs: "EC2 Instance Connect", "Session Manager" (which is selected), "SSH client", and "EC2 serial console". Under the "Session Manager" tab, the heading "Session Manager usage:" is followed by a bulleted list of instructions. At the bottom right, there are "Cancel" and "Connect" buttons.

EC2 > Instances > i-0731396381ba172c6 > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0731396381ba172c6 (nmbk-dev-environment-ide) using any of these options

EC2 Instance Connect | **Session Manager** | SSH client | EC2 serial console

Session Manager usage:

- Connect to your instance without SSH keys or a bastion host.
- Sessions are secured using an AWS Key Management Service key.
- You can log session commands and details in an Amazon S3 bucket or CloudWatch Logs log group.
- Configure sessions on the Session Manager [Preferences](#) page.

Cancel **Connect**

The screenshot shows a terminal window with a black background and white text. The session ID is "michael.klein@northwestern.edu-0331c3b7fae2c0c48" and the instance ID is "i-0731396381ba172c6". A "Terminate" button is visible in the top right corner. The terminal shows the user running "sh-5.2\$ sudo -liu ec2-user" and "Already up to date.". The user then runs "cd .nml-rdc-devtools" and "git:(main) ls -la". The output shows a directory listing with 4 files: "total 4", "drwxr-xr-x. 1 ec2-user ec2-user 60 Dec 1 21:51 .", "drwx----- 1 ec2-user ec2-user 1478 Dec 2 17:09 ..", "drwxr-xr-x. 1 ec2-user ec2-user 126 Nov 15 17:30 bin", and "drwxr-xr-x. 1 ec2-user ec2-user 204 Dec 2 17:09 .git". The user then runs "git:(main) bin/build" and the output shows a list of files: "app-environment* clean-s3* es-proxy* sg*", "backup-ide* dbconnect* https-proxy*", and "bin/build".

```
Session ID: michael.klein@northwestern.edu-0331c3b7fae2c0c48 Instance ID: i-0731396381ba172c6 Terminate
sh-5.2$ sudo -liu ec2-user
Already up to date.
+ cd .nml-rdc-devtools
+ .nml-rdc-devtools git:(main) ls -la
total 4
drwxr-xr-x. 1 ec2-user ec2-user 60 Dec 1 21:51 .
drwx----- 1 ec2-user ec2-user 1478 Dec 2 17:09 ..
drwxr-xr-x. 1 ec2-user ec2-user 126 Nov 15 17:30 bin
drwxr-xr-x. 1 ec2-user ec2-user 204 Dec 2 17:09 .git
drwxr-xr-x. 1 ec2-user ec2-user 90 Nov 3 19:23 helpers
-rw-r--r--. 1 ec2-user ec2-user 2219 Nov 16 19:07 README.md
drwxr-xr-x. 1 ec2-user ec2-user 160 Oct 21 19:38 scripts
+ .nml-rdc-devtools git:(main) bin/build
app-environment* clean-s3* es-proxy* sg*
backup-ide* dbconnect* https-proxy*
```

Secret Sauce I: Automated Startup

```
$ ssh bob.dev.nu1rdc.northwestern.edu
```

```
# ~/.ssh/config
```

```
Host *.dev.nu1rdc.northwestern.edu
```

```
User ec2-user
```

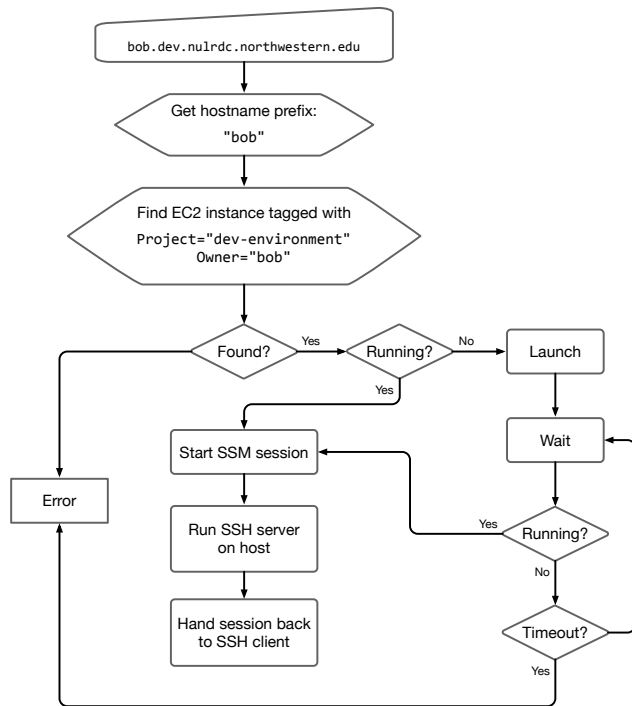
```
ForwardAgent yes
```

```
ProxyCommand sh -c "~/.ssh/nu1-ssm-proxy.sh %h %p"
```

Works regardless of how the SSH connection is initiated:

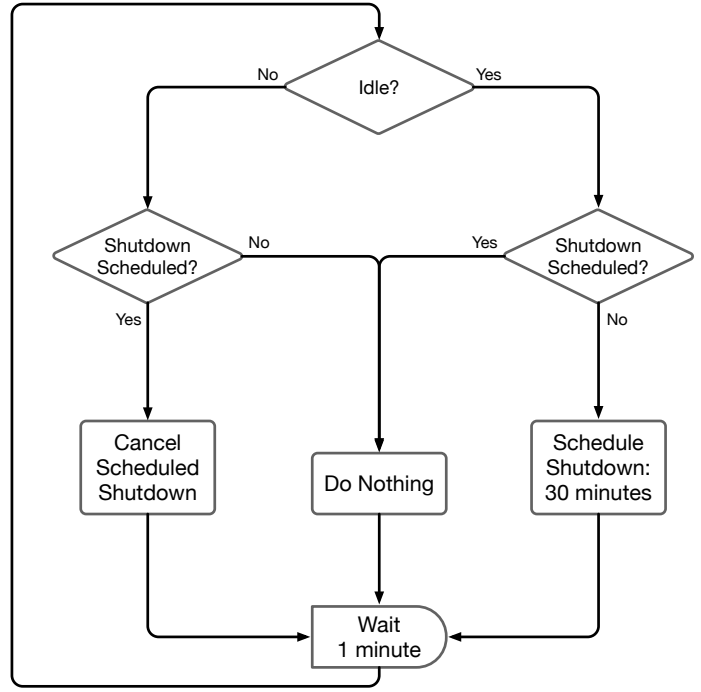
- Terminal Session: ssh, scp, rsync, or sftp
- VS Code Remote-SSH Extension
- Database Client Proxied Connection Configuration

Typical cold start time: 45 seconds



Secret Sauce II: Automated Shutdown

- Definition of “idle”
 - No “keep-alive conditions”
- Definition of “keep-alive condition”
 - Active VS Code Server
 - Active tmux session
 - ~/.keep-alive file



Secret Sauce III: App Configuration

- Instance tag (bob) and app environment (dev/test)
- [direnv](#) / .envrc
- AWS Secrets Manager
 - Some apps built to use secrets directly
 - Others require helper scripts to populate the environment

Secret Sauce IV: Addressing

- On startup: Register dynamic IP address as `bob.dev.nu1rdc.northwestern.edu`
- Temporarily open specific firewall ports with `$ sg open CIDR PORT`
- Dynamic web proxying with `$ https-proxy REMOTE_PORT LOCAL_PORT`

How did we do it?

First pass: AWS Cloud9 IDE

- Complicated bootstrapping script + Terraform
- Provided the bones of the automated startup and shutdown scripts
- Limited options for OS, volume size, other EC2 instance features

April 2022

September 2022

Second pass: Fully Custom

- Terraform provisioning of all shared and individual resources
- EC2 “first boot” script installs all dependencies and tools
- Startup and shutdown scripts tailored to our specific needs
- Can start with any base image (currently using Fedora 36)

Backup & Restore: Backup

- Save list of installed asdf plugins & versions
- Save list of installed VS Code extensions
- Write VS Code version & build info
- Copy VS Code settings.json
- Create tarball of all symbolic links under \$HOME
- Tar up \$HOME to /tmp excluding symlinks, asdf, VS Code, caches, build artifacts, installed packages, other ephemeral files
- Copy tarball to shared S3 bucket

Backup & Restore: Restore

- Download user's tarball from shared S3 bucket
- Extract backup into \$HOME
- Reinstall correct build of VS Code using saved manifest
- Reinstall VS Code extensions using saved list
- Copy VS Code settings.json back into place
- Reinstall asdf plugins & tools using saved manifest
- Extract symlinks.tar.bz2 into \$HOME

Additional Features

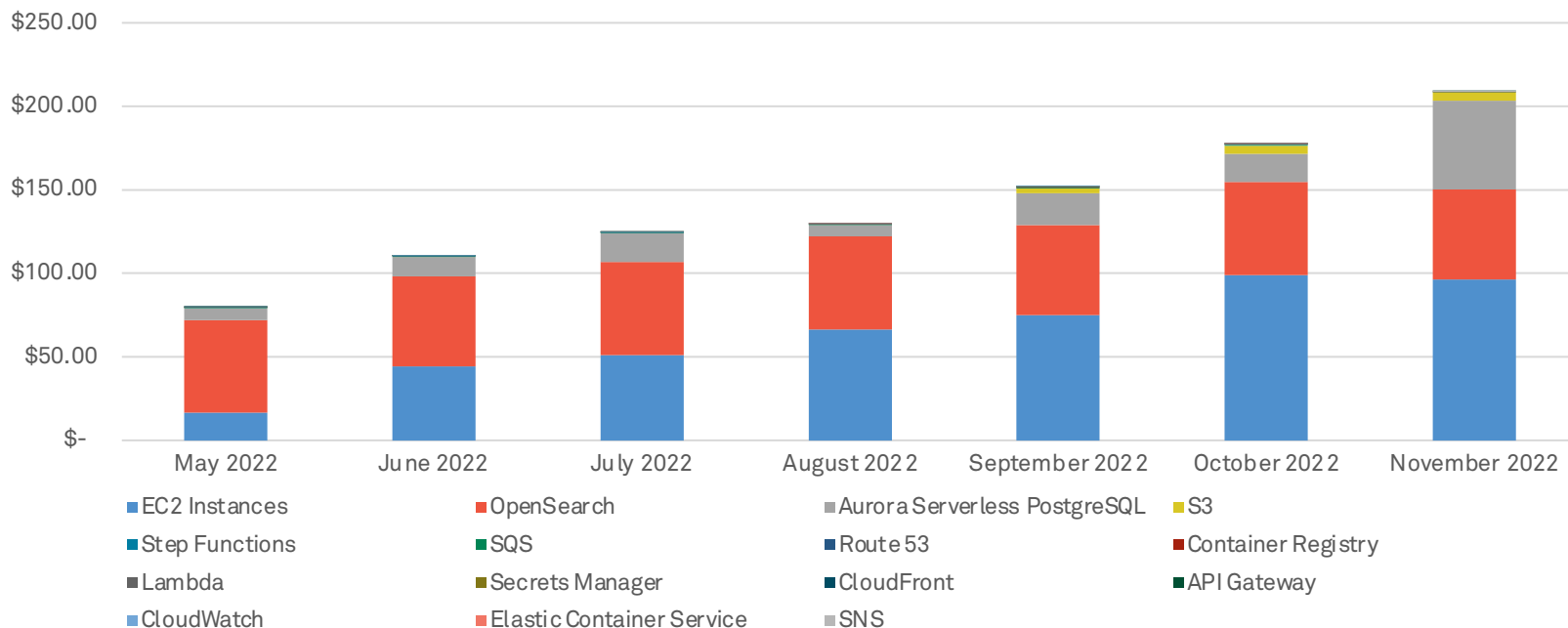
- Aurora Serverless database spins up and down in response to demand — no charges while idle
- Owner's public SSH keys installed from GitHub profile
- SSH key forwarding
- Scripts & tools are updated via `git pull` on every launch
- Environment configuration via AWS Secrets Manager

Caveats & Quirks

- No cure yet for “Developer forgets to shut down VS Code on Friday afternoon”
- No focus on security / privacy between developers
- SSH Agent purges keys after two hours
- Can't reconnect mid-shutdown
- VS Code sometimes fails to connect on the first try
- VS Code SSH-Remote plugin is proprietary
- OpenSearch Cluster doesn't auto-scale, and is slow to start up and shut down, so its 7½¢/hr. is a constant, 24x7 cost
- Some AWS features come with unanticipated costs
 - Most are predictable, but VPC configuration can surprise you!

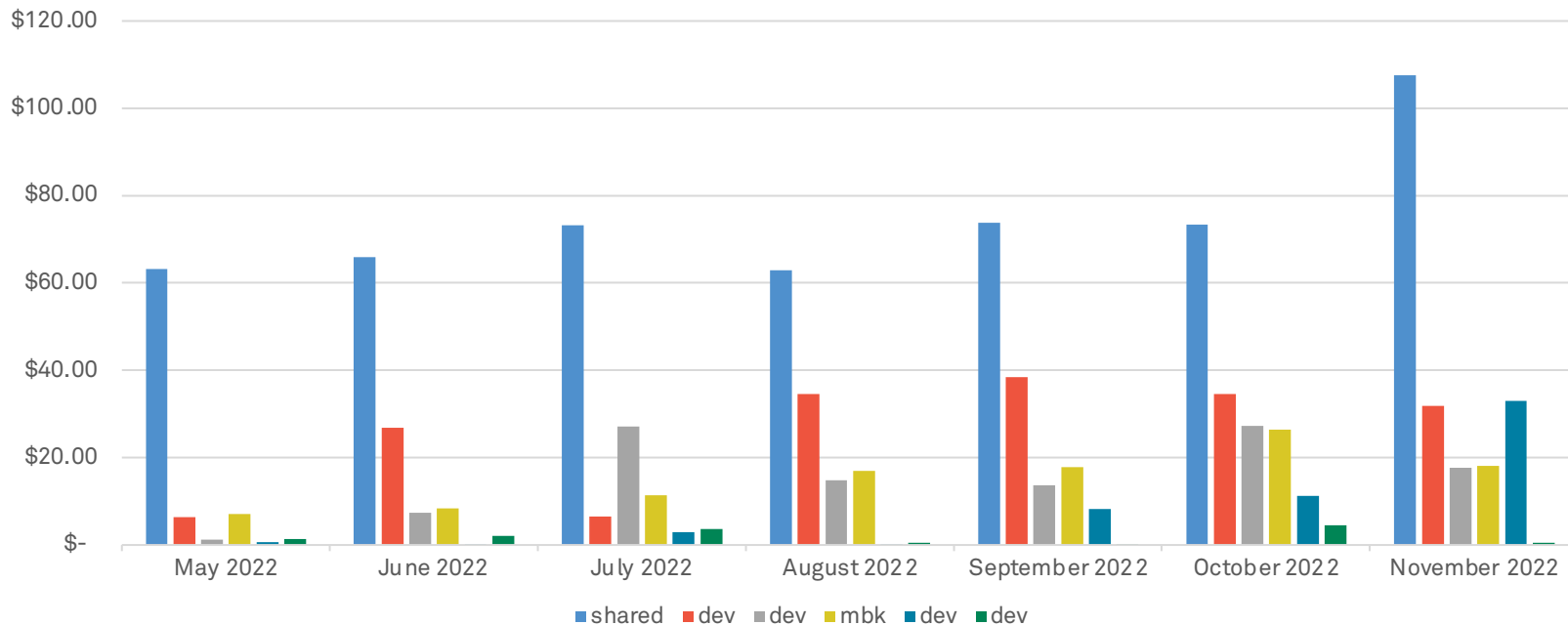
But what does this all cost?

Dev Environment Costs per Month by Service



But what does this all cost?

Dev Environment Costs per Month by Owner



Still to Come

- Better code organization
- Modular, reusable Terraform
- Improved maintenance utilities
- Better reporting tools
- Individual uptime alerts

Code & Contact

<https://github.com/nulib/aws-developer-environment>

Michael B. Klein (he/him/his)

Email: michael.klein@northwestern.edu

GitHub: <https://github.com/mbklein>