



A large, abstract network graph is visible in the background, consisting of numerous small, semi-transparent blue dots connected by thin white lines, creating a sense of a complex, interconnected system.

2022

# OPEN SOURCE IN FINANCE FORUM

## New York

PRESENTED BY





# Loosely Coupled Micro-Frontends And Capital One's Contact Centers

Stephen Husak  
Distinguished Engineer

Noah Mandelbaum  
Distinguished Engineer

# Agenda

- From Monolith to Micro-Everything
- Implementation Details
- Choices and Learnings



"Daisy" - Photo Source/Credit: Patty Edds, Sr. Risk Manager, Technology

# Who Are We?



Steve Husak  
Distinguished Engineer

Joined Capital One in 2014.

Greenfield architectures, sound engineering principles, easy-to-use developer experiences



Noah Mandelbaum  
Distinguished Engineer

Joined Capital One in 2012.

Architecture, technical teamwork.

# Disclaimer

- You will see a lot of dogs in this presentation.
- We are inclusive to all pets.
- All the dogs here within are part of our Capital One family!



"Turkey" - Photo Source/Credit: Steven Black, Business Analyst, Retail Bank

# Our Monolith Emerged About 15 Years Ago

- The new system had to allow more than 20,000 contact center associates to help customers.
- We built it quickly - we had to replace an older contact center system due to contractual obligations.
- Capital One chose .NET WebForms, ASP.Net and C# running on Windows Servers - just about everything was server-side.



"Sawyer" - Photo Source/Credit: Christy Mazza, Principal Project Manager, Capital Markets

# Running The Monolith Could Be Exhausting

- Legacy Infrastructure
  - 100+ on-premise servers to manage.
- Build/Test Cycle Was Slow
  - Developer experience was painful
  - Builds took a full day.
  - Testing took days.
- Large Batch Delivery
  - 1-2 releases a month with 100s of changes with hundreds of software engineers contributing to a single codebase.
  - Difficult to back out mistakes.
- Large Failure Blast Radius
  - Application was stateful and fault tolerance was suboptimal.
  - Many direct connections to data sources.



"Chloe" - Photo Source/Credit: Paula Kiley-Gerdes, Manager - Project Management, Retail Bank

# Nobody Was Very Happy With The Monolith

- Different lines of business wished to release on their own cadences.
- Complex negotiations were required to make it all work.
- Our contact center agents could do their job, but they wanted the software engineers to fix bugs faster and make improvements faster.



"Shuri" - Photo Source/Credit: Alicia Neumann, Principal Associate, Marketing

# New Ways Of Thinking Caught Our Eyes

- 2010 - the Continuous Delivery book was published.
- 2012 - Capital One began to embrace APIs.
- 2013 - Capital One started their public cloud journey with AWS (finished in 2020).
- 2014-2015 - the Capital One engineers who worked on the monolith began experimenting with SPAs and learned more about Node.js.



"Princess" - Photo Source/Credit: Noah Mandelbaum, Distinguished Engineer, Card

# Micro-Frontend Architecture (2016) Really Excited Us

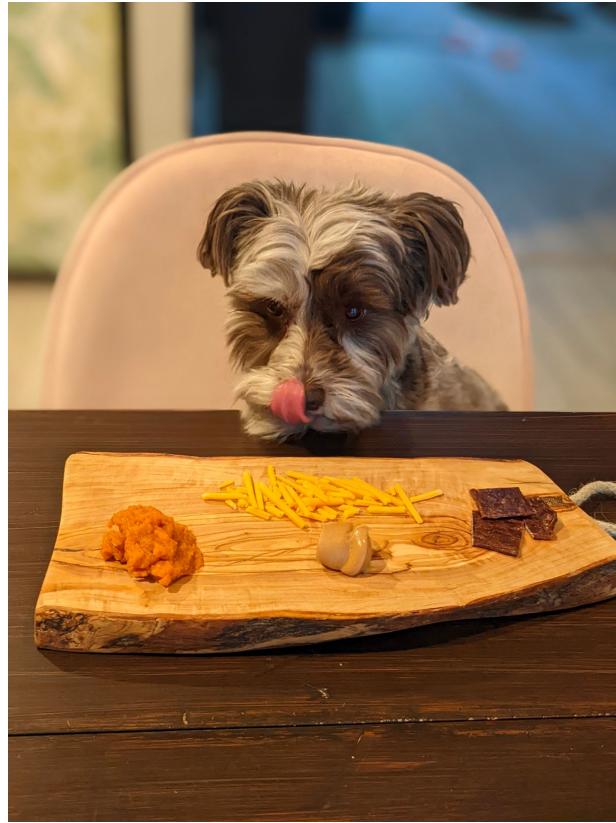
- Model based on business domains.
- Hide implementation details
  - be loosely coupled and contract-based in communication.
- Isolate failure.
- Decentralize as much as possible.
- Release independently.



"Ruby" - Photo Source/Credit: Christine Durlak, Admin Assistant, Enterprise Architecture

# We Wanted It All!

- Clear lines of ownership to minimize organizational friction.
- The ability to deploy and release at any time - with little toil.
- Limited failure blast radius if a part of our platform encountered an error.
- Smaller, simpler codebases that developers could quickly understand.
- Room for our software engineers to iterate incrementally



"Presley" - Photo Source/Credit: Annette Bonacci, Principal Process Manager, Retail Bank

# But Migration to MFEs Was Not Straightforward

- It would be wrong to say that any of us knew exactly what we were doing up front.
- We iterated multiple times - the new platform saw at least five major pivots as we built and discarded ineffective models.
- We had to have same functionality of the large legacy contact center application.
- Our product managers also wanted to introduce business process innovations while we were migrating.



"Luigi" - Photo Source/Credit: Rita Dilorio, Lead Software Engineer, Technology

# We Learned We Needed A Good Foundation

- A single unifying design system that allowed the platform to create the illusion of a “single application”
- A standard CI/CD pipeline that automated everything.
- Open governance/knowledge shared among the groups that participated in the platform.
- Constant measurement of developer experience and end user experience.



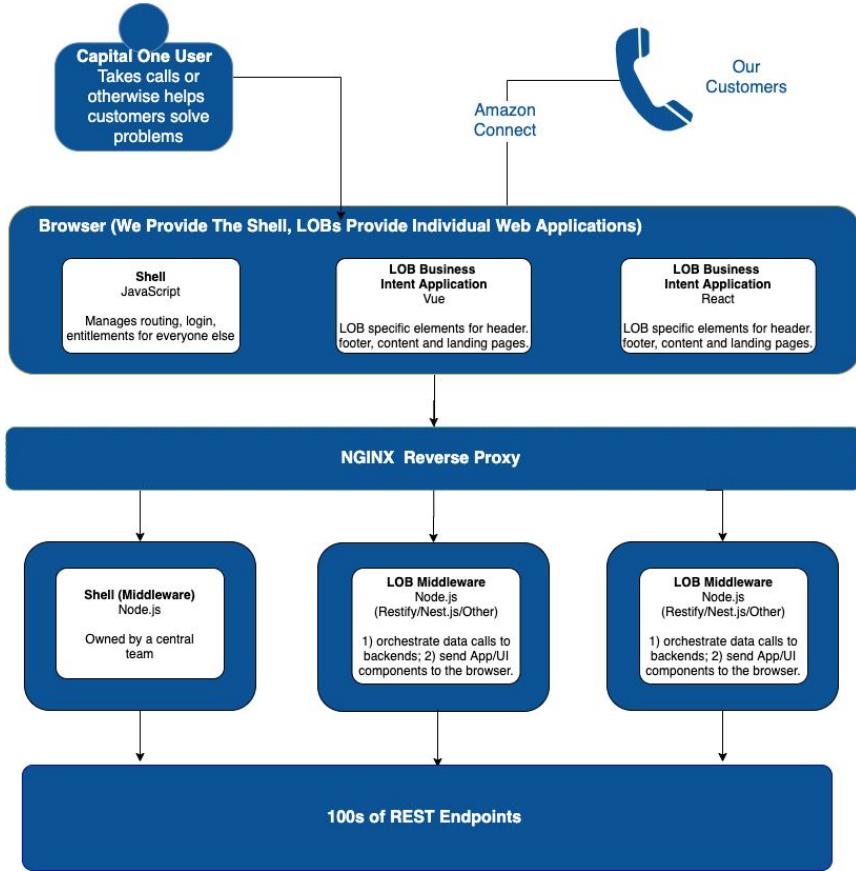
“Hazel” - Photo Source/Credit: Alex Hardman, Solution Architecture, Commercial Bank

# Where We “Sit” Today

- Reduced time to market with no outages
  - The legacy monolith averaged 1-2 releases a month.
  - Recently, the micro-frontend platform:
    - Had several dozen teams contributing at the same time.
    - Carried out hundreds of independent releases (~dozen releases per day).
    - Achieved low change failure rate - no outages were associated with the increased release cadence.
- Highly decomposed system that encourages incremental change
  - More than 50 micro-frontends and a similar number of independent Node.js services on the backend.
  - Bugs can frequently be resolved in hours without heroics.
  - 100% Cloud native.
- Very good developer experience
  - Our internal surveys shows our developers really like working in our ecosystem, compared to comparable systems in our organization.

# An App Shell With Multi-Level Routing

- Our federated model allows teams to choose the technologies that suit their needs while giving the customer a “single app” experience.
- Production libraries we use include:
  - Fastify
  - Pino
  - NestJs
  - React
  - Restify
  - Undici
  - Vue.js
- Development dependencies include:
  - Cypress.io
  - Jest
  - Mocha/Sinon/Chai
  - Mountebank



# Routing by convention

/ tenant / domain / container / app / resource

Tells the system what configuration to use for page composition

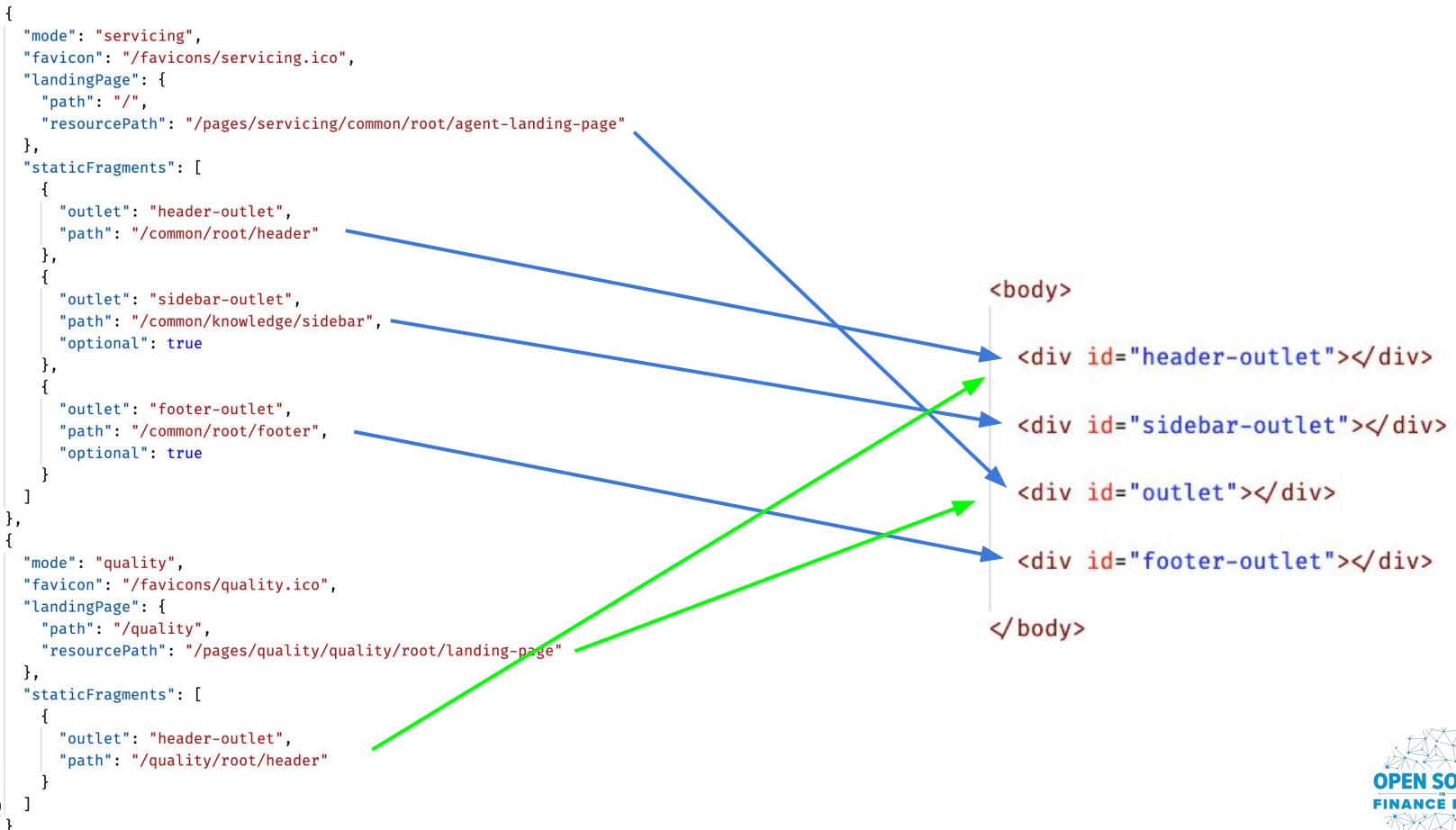
Opaque organizational unit (business domain)

Coarse-grained group of functionality

Micro-UI code

Resources of the application

# Configuration-based page composition keeps things flexible.



# The end result is a cohesive application in the browser.

Demo - talk\_application\_main

localhost:3000/pages/talk/talk/application/main

**HEADER**  
This is a subheader in the header app

**This is the application page**  
This is the subheader on the application page

**Some Content**

Your input:  
An Input  
Input Placeholder

**Sidebar**  
Context-sensitive is displayed here as needed

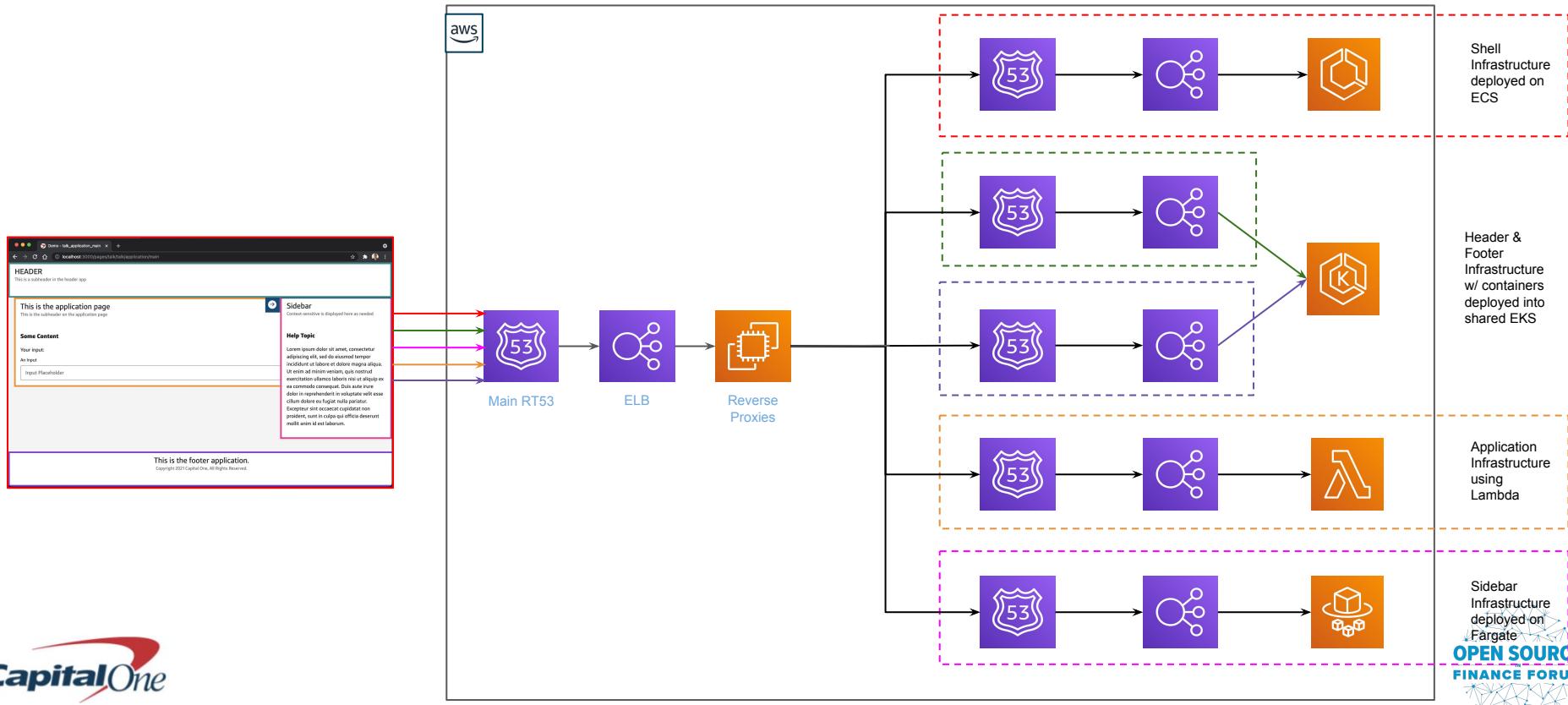
**Help Topic**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

**This is the footer application.**  
Copyright 2021 Capital One, All Rights Reserved.

**Capital One**

# A reverse proxy brings it all together under one domain and allows for flexible hosting solutions.



# Lessons Learned along the way

- Provide A Good Developer Experience
- Limit Cognitive Load Using Node.js
- Decide on Mono Versus Poly
- Define A Support Model
- Never Stop Refactoring



"Louie" - Photo Source/Credit: Brooke Simone, Project Manager, Legal

# Lesson Learned - Provide A Good Developer Experience

- Monolith required developers to build & run everything through IIS Server (remember it was ASP.Net based).
- Developers now only have locally what is needed for their development
- System runs in native Node.js processes and/or Webpack dev servers
- A “developer” proxy brings it all together
- We have a full set of maintained documentation with tutorials, how-tos and reference pages



“Scamp” - Photo Source/Credit: MaryAnne Gresham, Sr. Manager - Software Engineering, Card

# Lesson Learned - Limit Cognitive Load Using Node.js

- Full-stack JavaScript simplifies the developer experience greatly
- Context switching has documented effects on developer productivity. Sticking to one language helped remove another forced context switch to developer work.
- Code, tools and testing patterns could be shared between the frontend and the backend.
- So far, no team has opted for using JavaScript on the frontend and another language on the backend (Java, Go).



"Wksi" - Photo Source/Credit: Heather A. Hosmer, Sr. Manager - Senior Counsel, Legal

# Lesson Learned - Decide on Mono Versus Poly

- This can be a tough topic - and it was in our space
- We like polyrepo because it preserves key characteristics of the microservice approach for UI applications:
  - Independent deployability
  - Modularity
  - Encapsulation
  - Clear ownership
- That is not to say this is the absolute right answer - you just have to consider the trade-offs



"Lewy" - Photo Source/Credit: Andy Littrell, Cafe Ambassador, Retail Bank

# Lessons Learned - Define A Support Model

- Internal customers wanted assurances that our core components will remain secure and bug-free, so they can focus on business intent.
- We created a support model in which “trusted contributors” dedicate time to new library features, comprehensive documentation and security patches.
- We also employ a N-1 versioning strategy - consuming teams are asked to stay current with dependencies (although this can be a challenge).



"Iris" - Photo Source/Credit: Caroline DeLoach, Agile Delivery Lead, Technology

# Lessons Learned - Never Stop Refactoring

- Be ready to refactor to best-in-breed tooling.
- As you get bigger, you have to think about the appropriate time to move to be secure and performant.
- Negotiate explicitly with your product partners as you move through the refactoring process.



"Winnie" - Photo Source/Credit: Larry Contratti, Sr. Manager, Design

# Final Notes

- Our journey is not unique but the path we took is.
- We've achieved what we needed for our platform currently, but there is always more work to be done.



"Cosmo" - Photo Source/Credit: Steve Husak, Distinguished Engineer, Commercial Bank

# Thank you!

## Noah Mandelbaum

LinkedIn: <https://www.linkedin.com/in/noahmandelbaum/>

Twitter: @NoahMandelbaum

Email: [noah.mandelbaum@capitalone.com](mailto:noah.mandelbaum@capitalone.com)

## Steve Husak

LinkedIn: <https://www.linkedin.com/in/steve.husak>

Twitter: @shusak

Email: [stephen.husak@capitalone.com](mailto:stephen.husak@capitalone.com)

Slides: <https://github.com/shusak/osff2022>



"Laini" - Photo Source/Credit: Steve Husak, Distinguished Engineer, Commercial Bank

# Open Source Software COMMITMENT TO COMMUNITY

Capital one made an “open source first” declaration in 2014 and that’s when we made our first contributions to the open source community.

- We sponsor FINOS, OpenJS, Python, Continuous Delivery and the Cloud Native Computing Foundations to help keep open source sustainable
- Capital One’s contributions to the open source community have been significant and we’ve released more than 40 of our own software projects
- We’ve invested for years to build the culture and governance required to be open source-first in a highly regulated industry

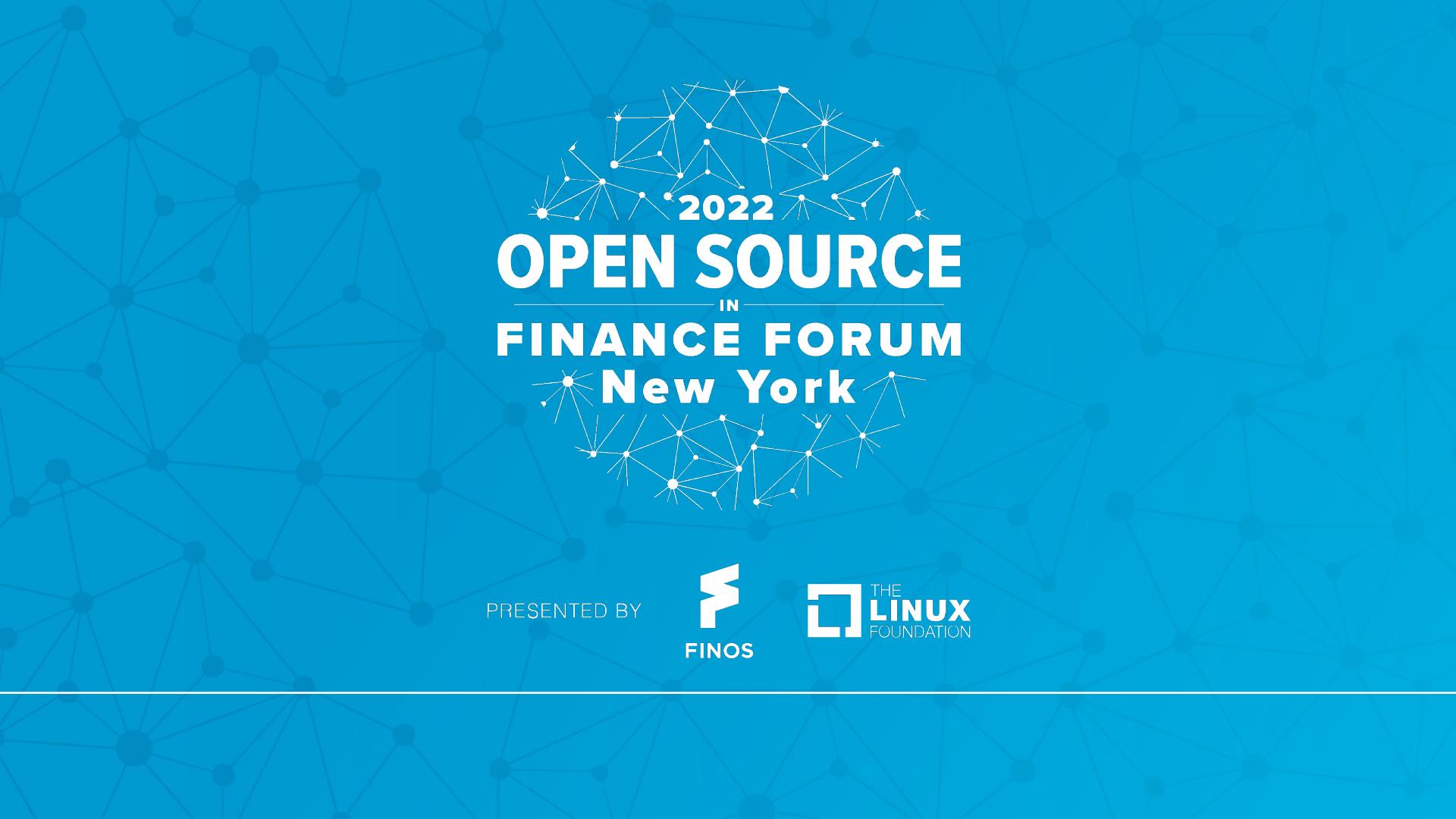


Featured Open Source Projects: Data Profiler, Rubicon-ML and Hygieia



# Want to learn more?

- Want to learn more about our Tech? Check out [Capital One Tech](#) to find out more about enterprise software solutions, ideas and stories.
- At Capital One, we celebrate and honor the differences that makes us all unique- inside and outside of work. Help us create a more equitable future for all! Join us! Visit [Capital One Careers](#) to view our open roles.
- Follow us on Twitter at [CapitalOneTech](#)



A large, abstract network graph is visible in the background, consisting of numerous small, semi-transparent blue dots connected by thin white lines, creating a sense of a complex, interconnected system.

2022

# OPEN SOURCE IN FINANCE FORUM

## New York

PRESENTED BY



# Appendix



# Capital One's technology transformation

