Monoliths to microservices: App Transformation

Hands-on Technical Workshop

API-led modernization

Why APIs

More than a technology, it's a way of doing business

- Your product is your business, both internal and external
- Entrepreneurial approach to solving problems and leveraging opportunities
- Customer service driven behaviors
- Support overarching organizational strategies with data-driven decisions

There is no “off” button for legacy applications

> When designing a new application you should design it in such a way as to make it easier for it to be strangled in the future. Let's face it, all we are doing is writing tomorrow's legacy software today. By making it easy to be strangled in the future, you are enabling the graceful fading away of today's work.
>
> - Martin Fowler

Application modernization

Adopting legacy applications to deliver value for modern business needs

Modernization patterns

Eliminate technical risk and speed execution

LIFT & SHIFT

- **Containerize** existing workloads
- Deploy them on a **PaaS**
- Keep external integrations and data on legacy
- Legacy applications have to be well written and suited

CONNECT & EXTEND

- **Legacy remains intact**
- New layer - new capabilities
- Deploy on **PaaS**
- **New integration points** between legacy and new layers **(Need for Agile Integration)**

RIP & REWRITE

- **Legacy is totally replaced**
- New interfaces and data
- Use **PaaS** to run
- Some data and features can be re-wrapped, but mostly are retired.

Modernization patterns

Eliminate technical risk and speed execution

LIFT & SHIFT

- **Containerize** existing workloads
- Deploy them on a **PaaS**
- Keep external integrations and data on legacy
- Legacy applications have to be well written and suited

CONNECT & EXTEND

- **Legacy remains intact**
- New layer - new capabilities
- Deploy on **PaaS**
- **New integration points** between legacy and new layers **(Need for Agile Integration)**

RIP & REWRITE

- **Legacy is totally replaced**
- New interfaces and data
- Use **PaaS** to run
- Some data and features can be re-wrapped, but mostly are retired.

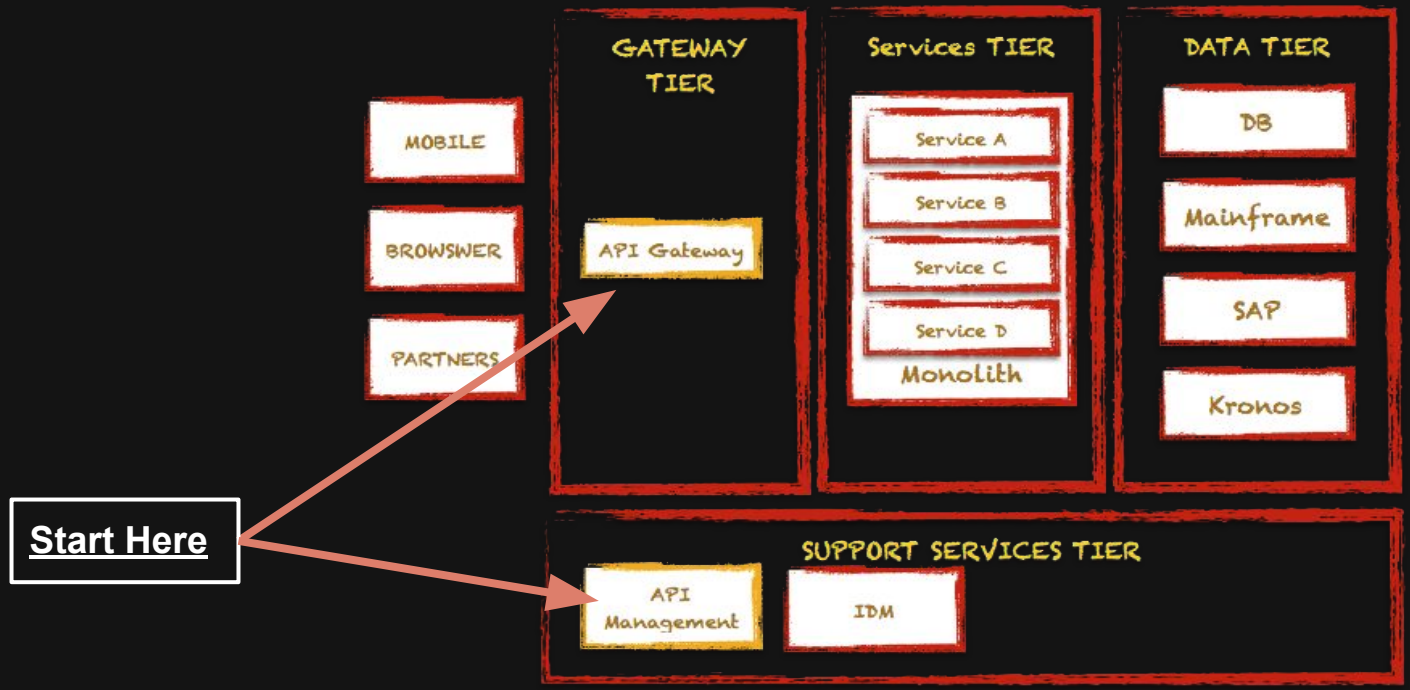
API-led modernization in practice: Connect and extend

AKA:

STRANGLING THE MONOLITH

Phase 1: Strangling the monolith

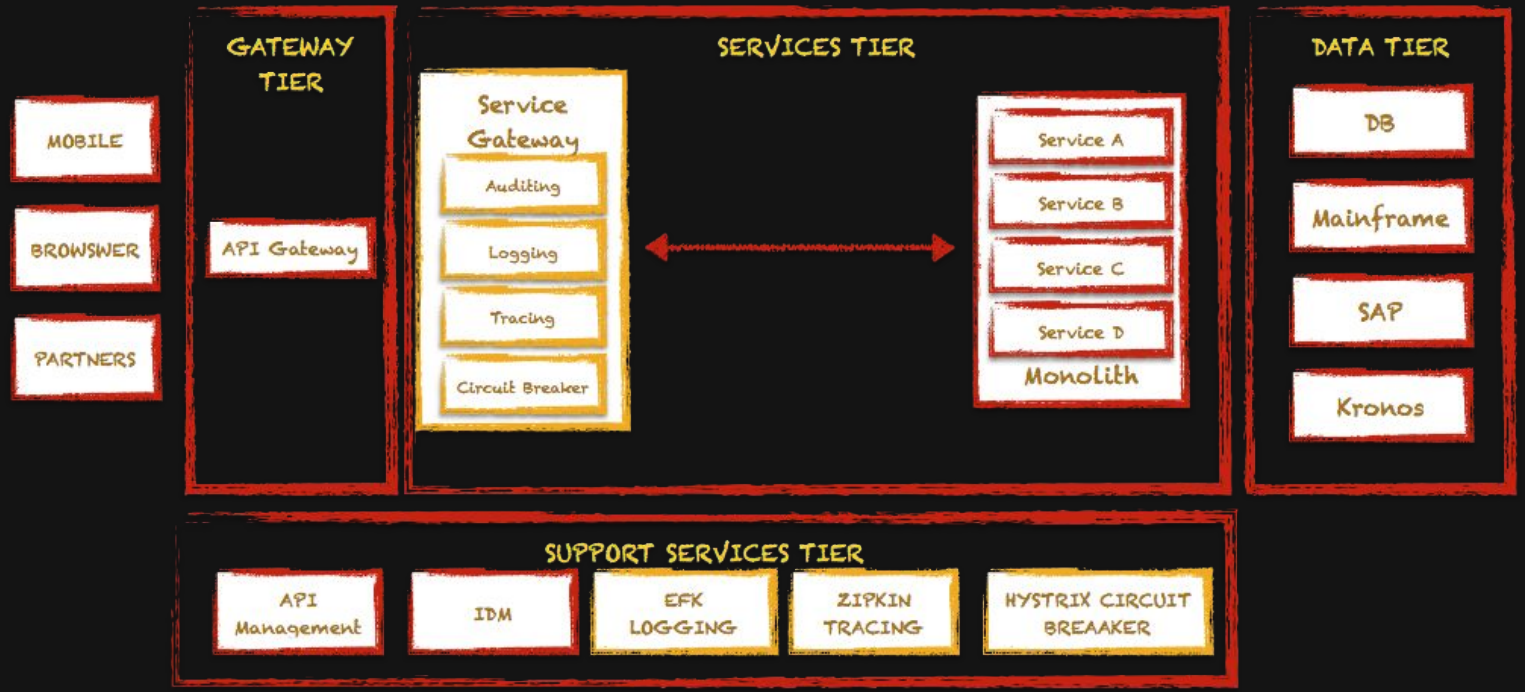
API First



Red Hat

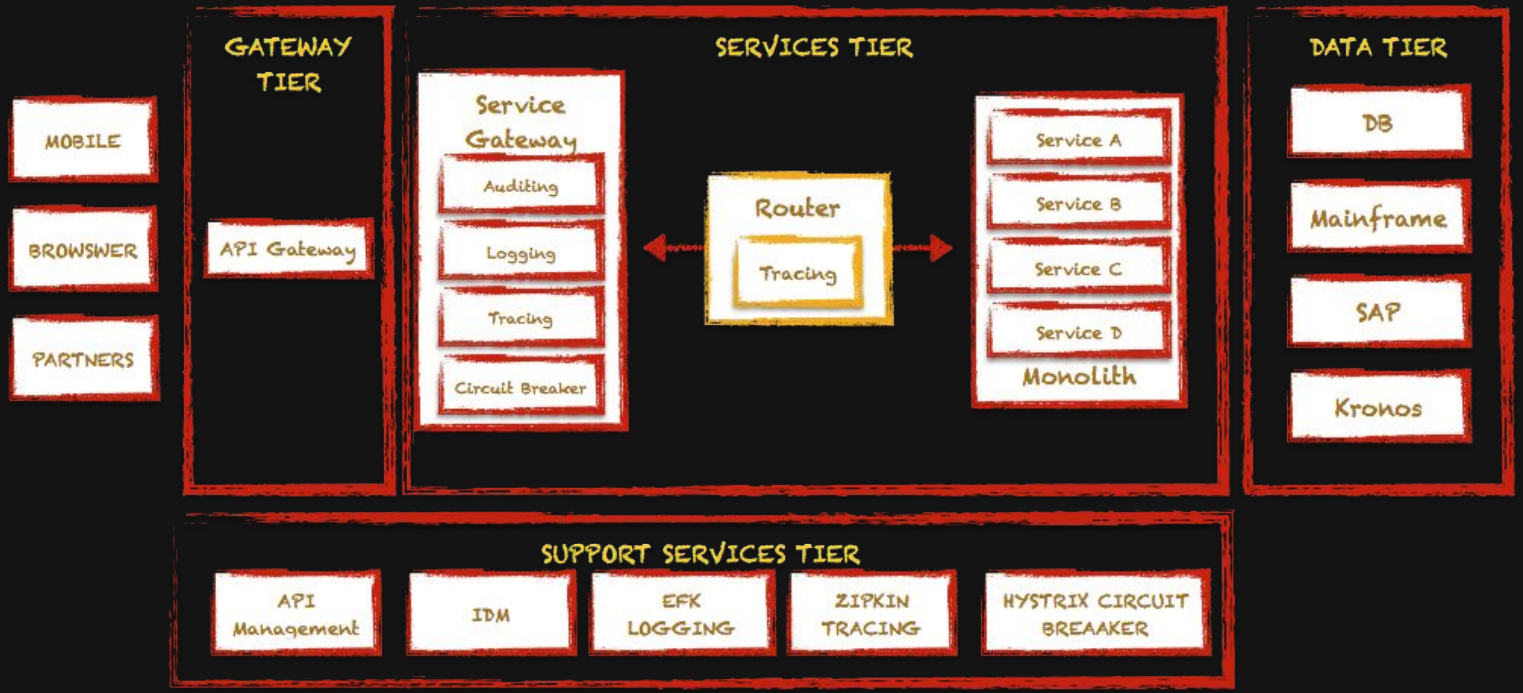
Phase 2: Strangling the monolith

Connect and extend with a services gateway



Phase 2: Strangling the monolith

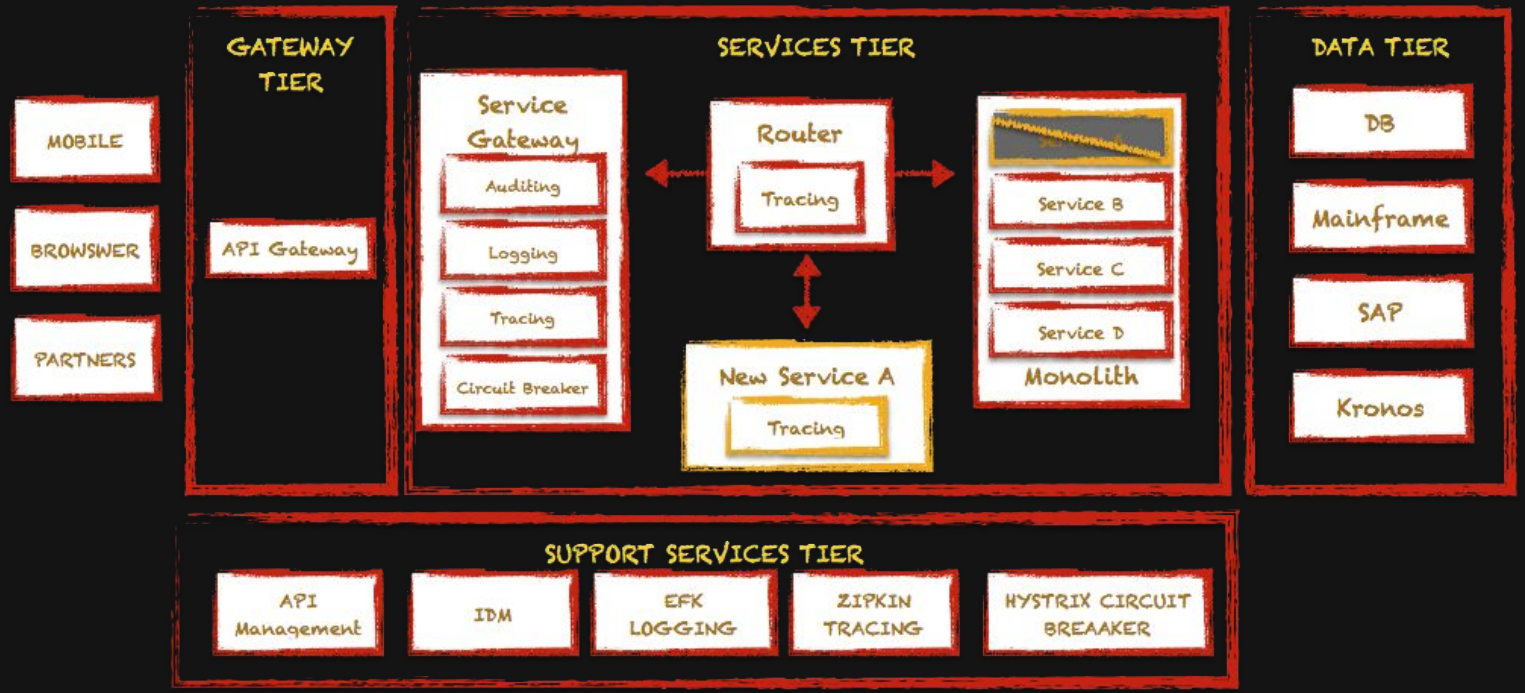
Create a path for new services



Red Hat

Phase 2: Strangling the monolith

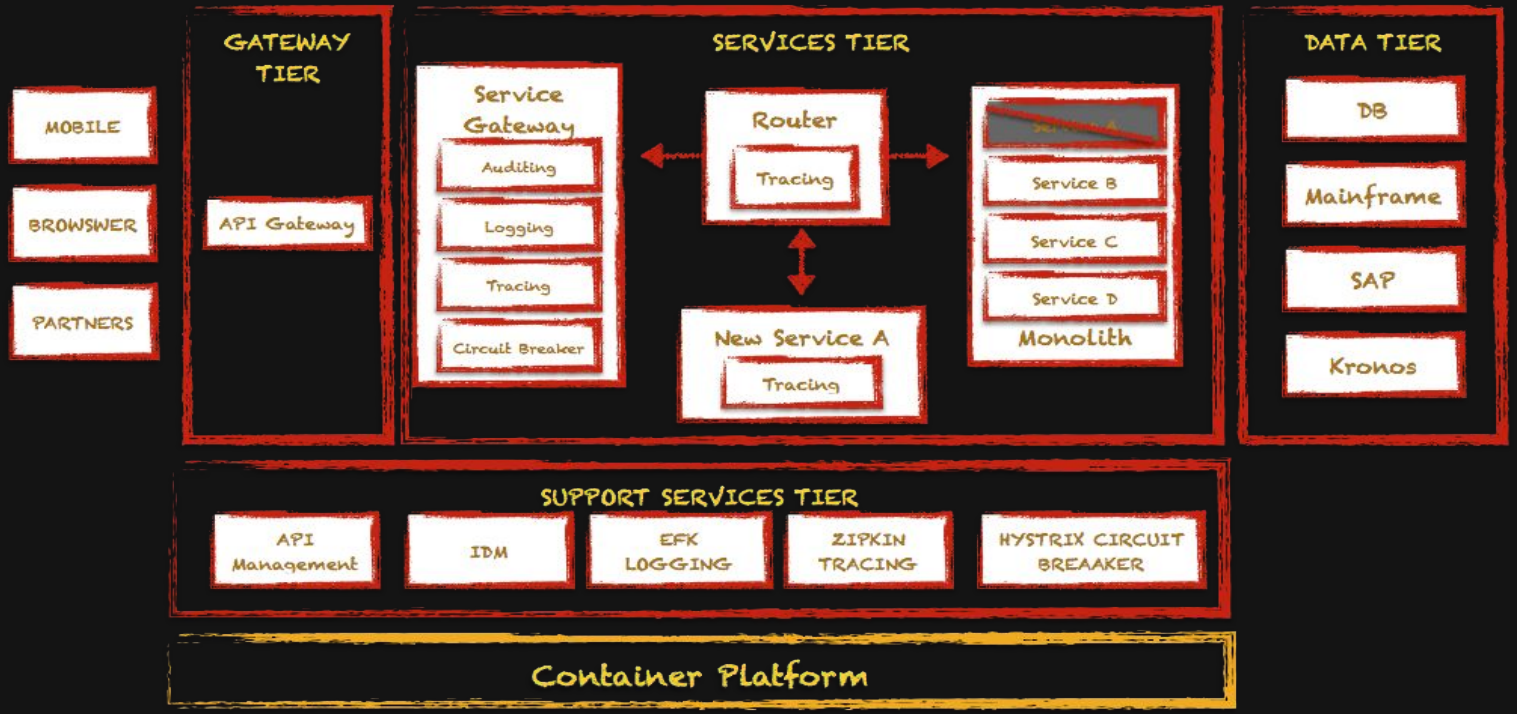
Out with the old, in with the new



Red Hat

Phase 3: Strangling the monolith

Containerization



Red Hat

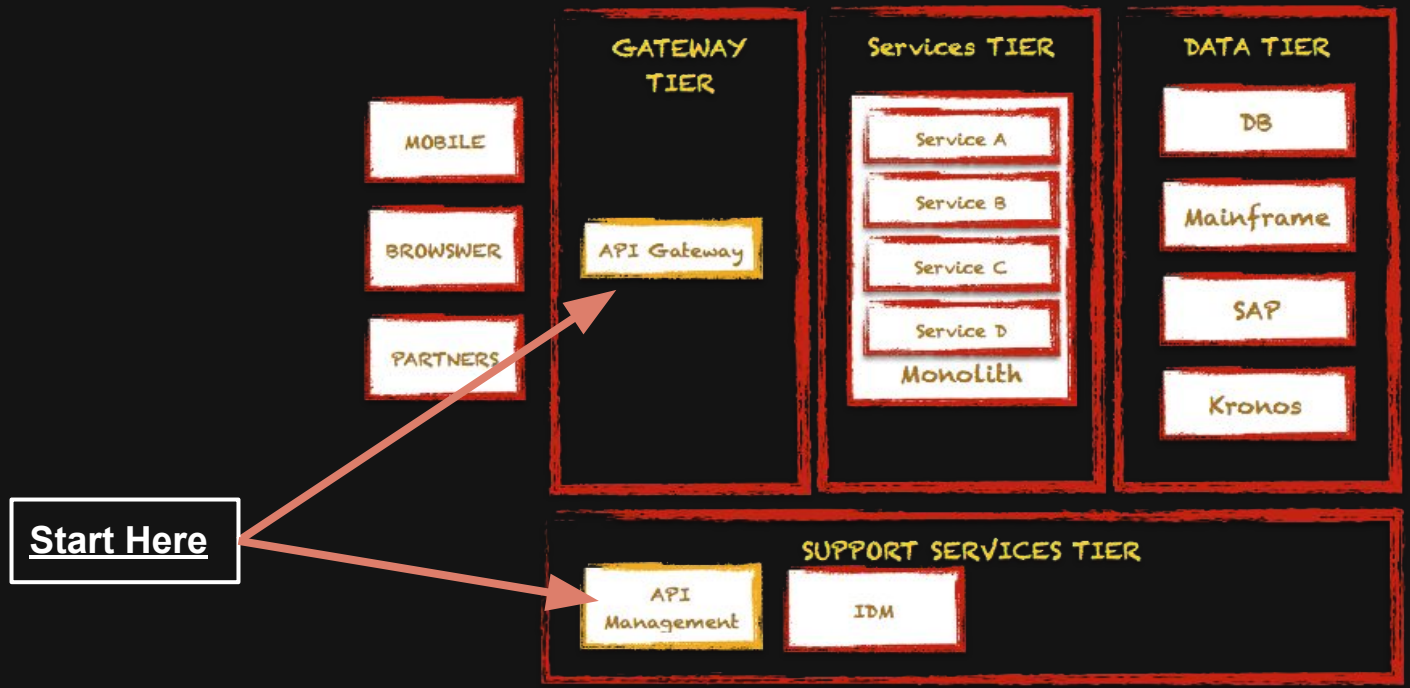
Drill down

Phase 1: Strangling the monolith

Exposing APIs

Phase 1: Strangling the monolith

Expose APIs



Red Hat

Red Hat 3scale API Management

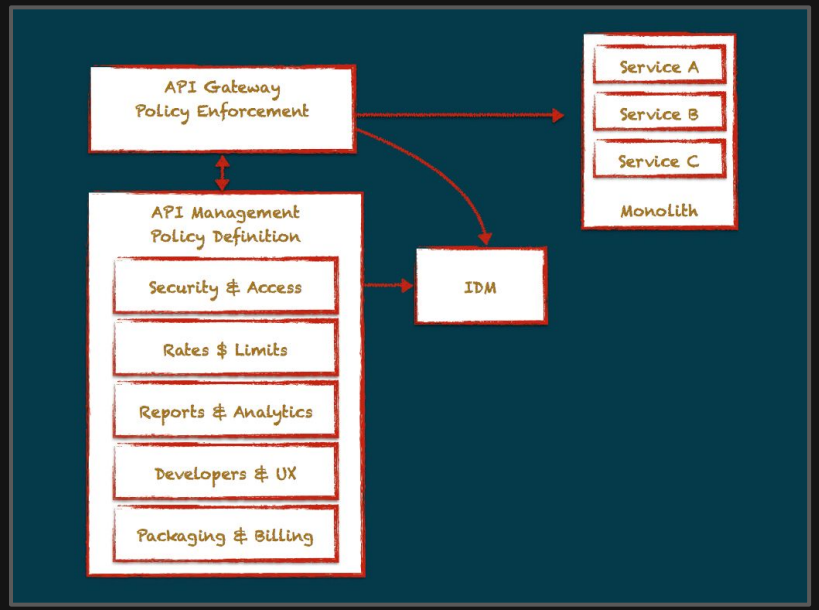
Red Hat

Why API Management?

API Management is where we start growing and learning

5 Components to API Management

- Security & Access Control
- API Contracts, Throttling & Rate Limits
- Reports & Analytics
- Developer & Partner UX
- Packaging, Billing & Payments

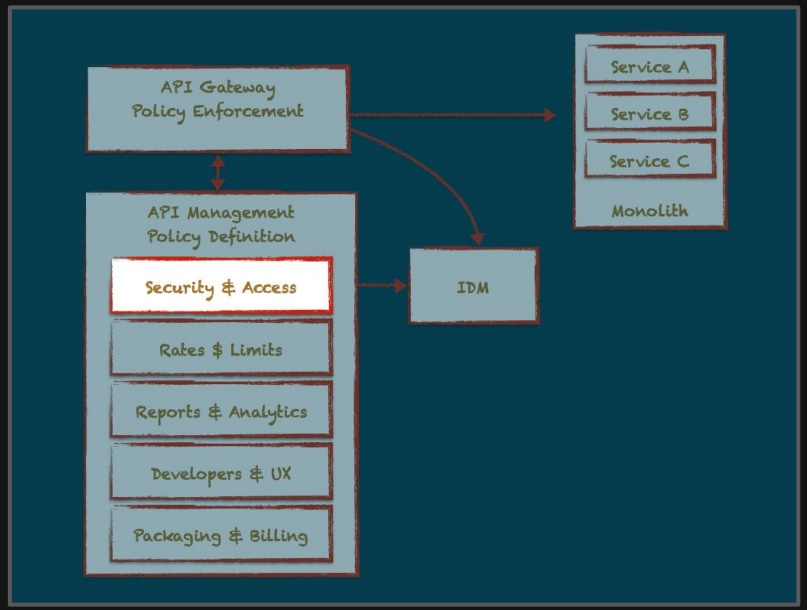


Red Hat

Why API Management?

API security and access control

- Authenticate traffic
- Restrict by policy
- Drop unwelcome calls
- Protect backend services
- Generate overage alerts
- Impose rate limits

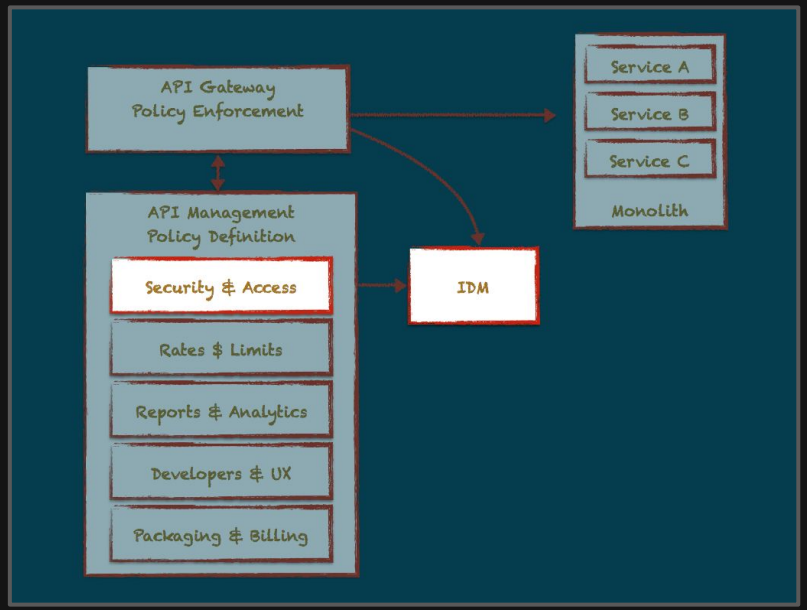


Red Hat

Why API Management?

API security and access control

- Multiple authentication mechanisms
  - API Key
  - APP ID / API Key
  - OAuth 2.0

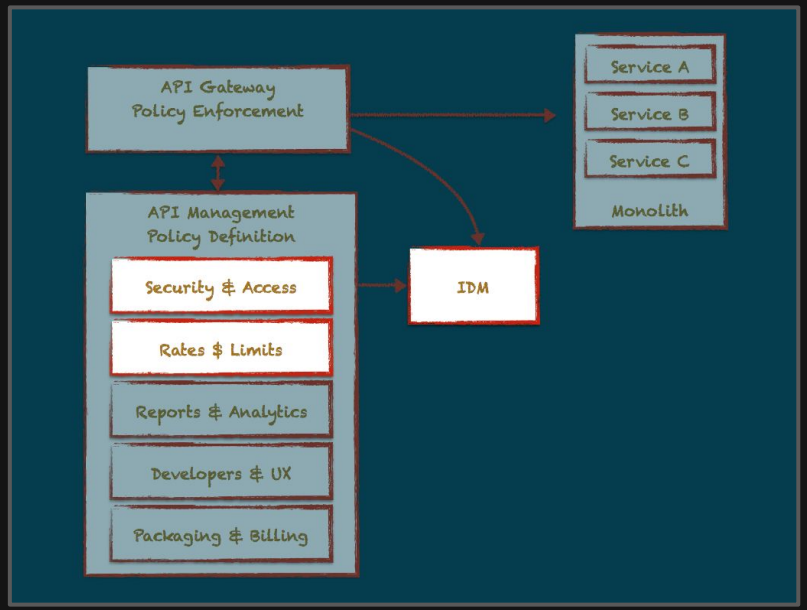


Red Hat

Why API Management?

API contracts, throttling and rate limits

- Allow/restrict access to your API endpoints along with rate limits
- Rate-limit accounts
- User and endpoint level

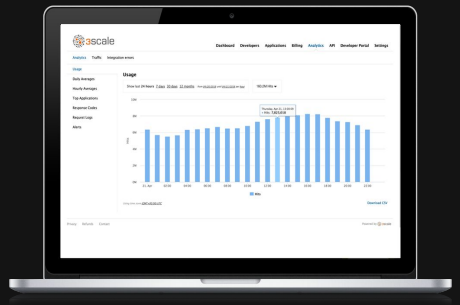


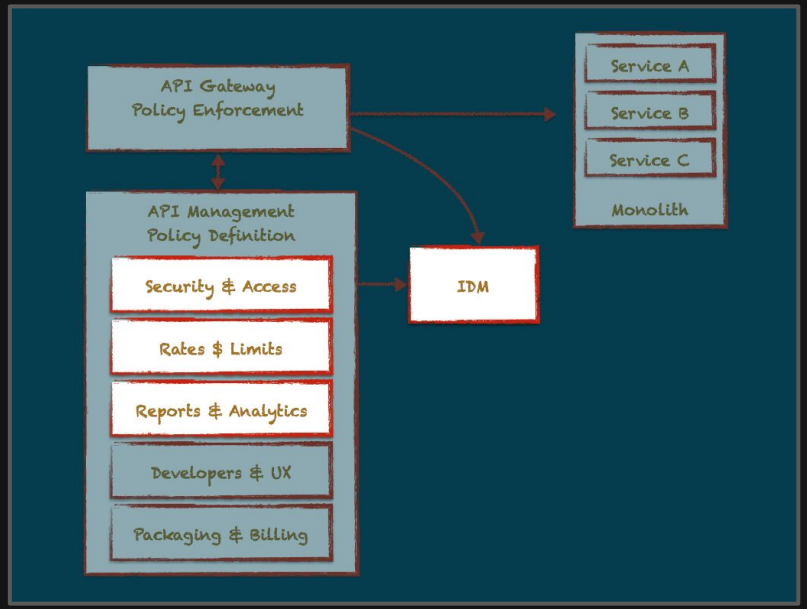
Red Hat

Why API Management?

Reports and analytics

- Analytics providing intelligence about the performance, and traffic patterns
- Visibility who is doing what where
- Automatic based on behaviors



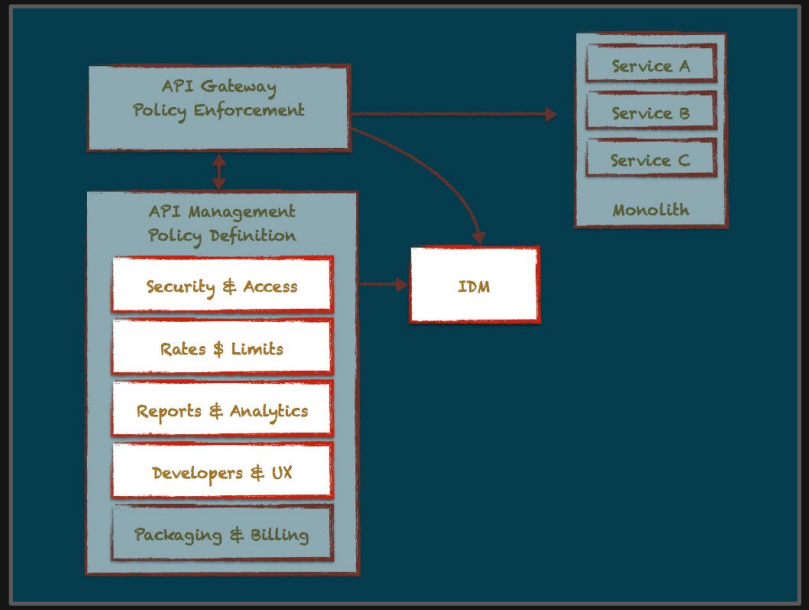


Red Hat

Why API Management?

Developer portal and user experience

- Your brand
- Your user experience
- Your user interface

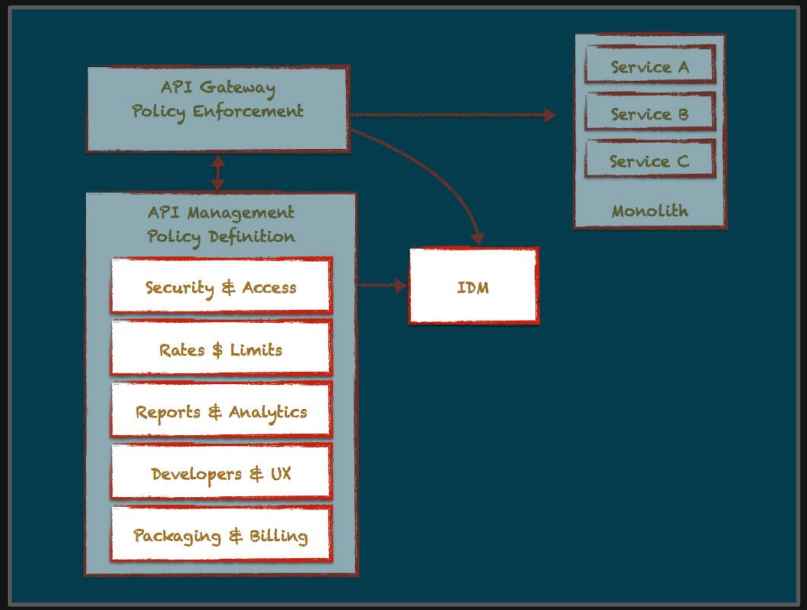


Red Hat

Why API Management?

Packaging, billing and payments

- Multiple pricing rules
  - One time payment
  - Fixed/Variable recurring fee
  - Cost per unit
  - Tiered pricing
- Billing cycles
- NO CC DATA STORED



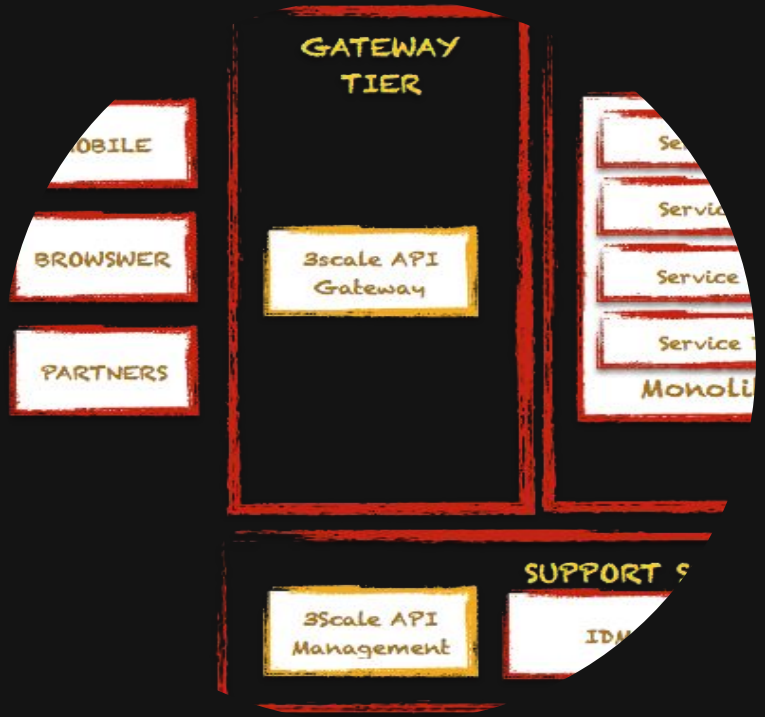
Red Hat

Red Hat 3scale API Management

Expose APIs faster with 3scale

Keys Differentiators

- Hybrid Cloud Architecture
- Separation of Concerns
  - Slim DMZ deployments
  - NO DATA AT REST
- 1st class citizen in an automated DevOps tool chain



Red Hat

Thank you

LinkedIn: linkedin.com/company/red-hat

YouTube: youtube.com/user/RedHatVideos

Facebook: facebook.com/redhatinc

Twitter: twitter.com/RedHatNews

Google+: plus.google.com/+RedHat

LinkedIn: linkedin.com/company/microsoft/

YouTube: youtube.com/user/MSCloudOS

Facebook: facebook.com/microsoftazure/

Twitter: twitter.com/azure

Azure Friday: channel9.msdn.com/Shows/Azure-Friday

Azure | Channel 9: channel9.msdn.com/Blogs/Azure

Red Hat app modernization program

Unified approach

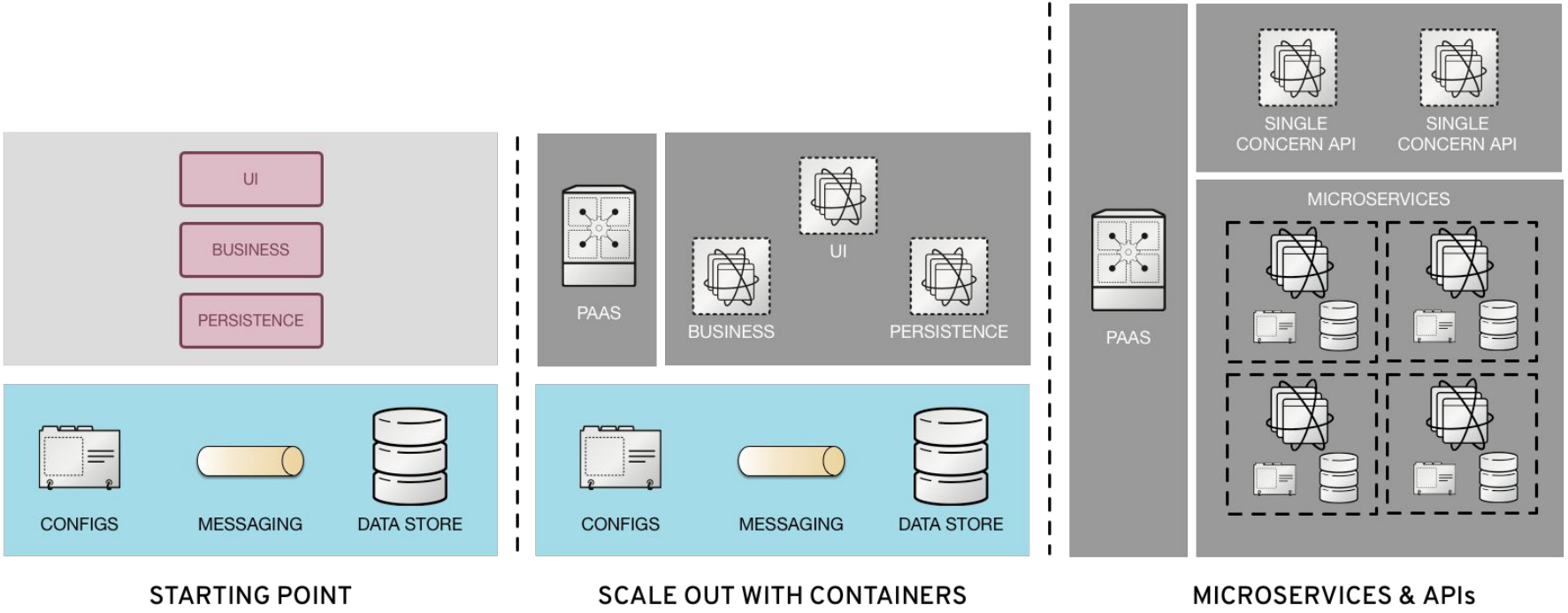
FIELD TESTED PATTERNS
Eliminate technical risk and speedup execution

MODERN SOFTWARE ENGINEERING
Methodical approach to evolve systems

MODERNIZATION FACTORY
Addresses all application types across portfolio

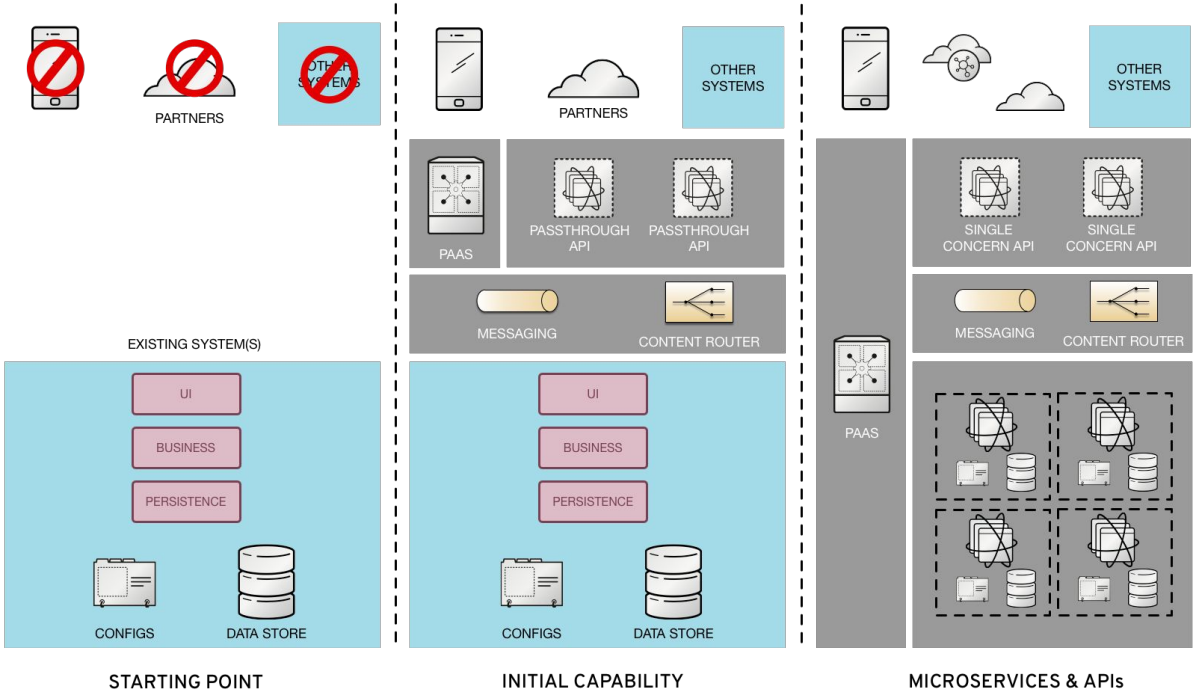
PURPOSE-BUILT TOOLS
Eliminate guesswork for predictable results

Lift & shift

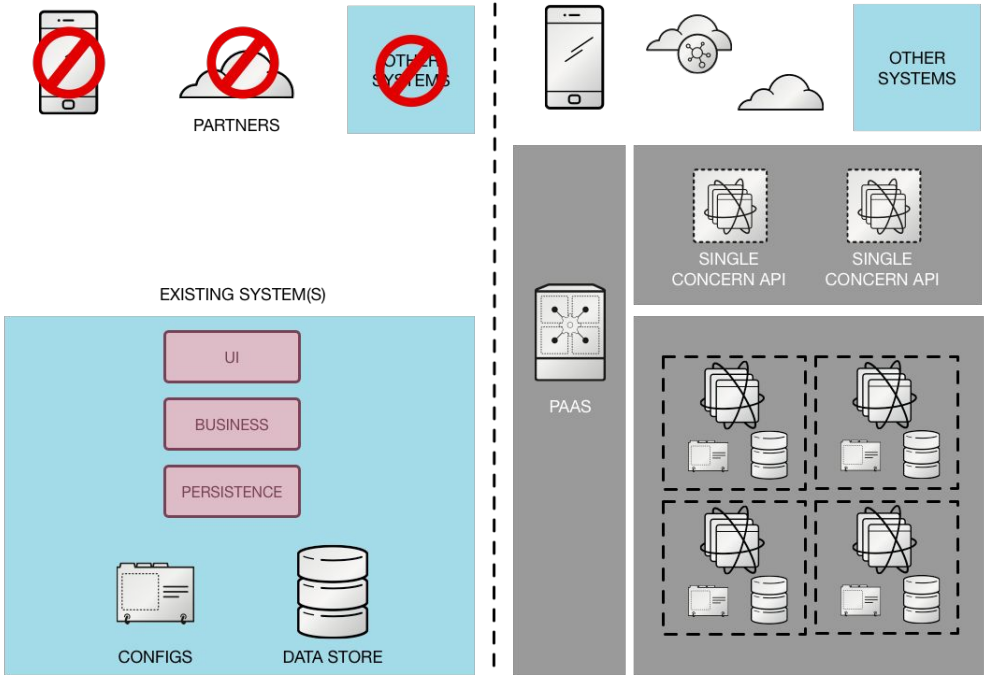


Red Hat

Connect & extend

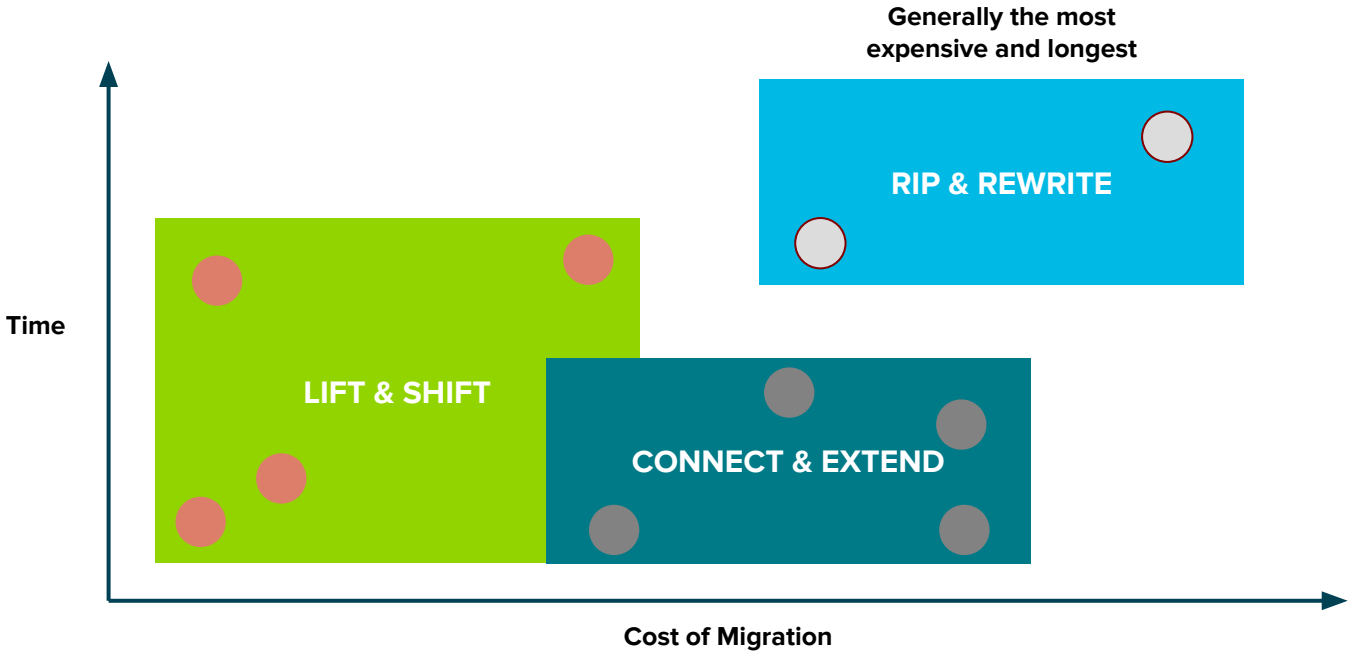


Rip & rewrite



Red Hat

How do they compare?



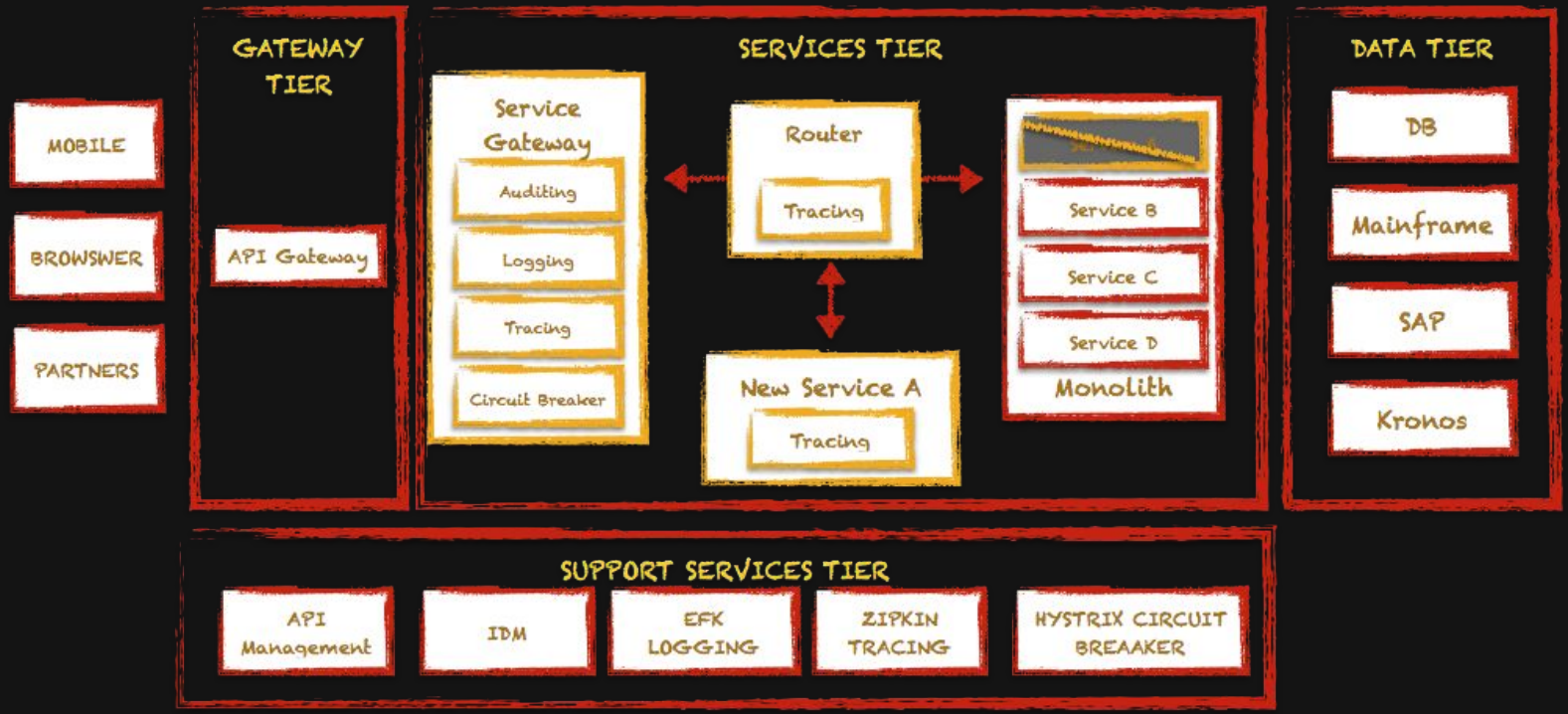
Drill down

Phase 1: Strangling the monolith

Developing APIs

Phase 2: Strangling the monolith

Developing our new APIs



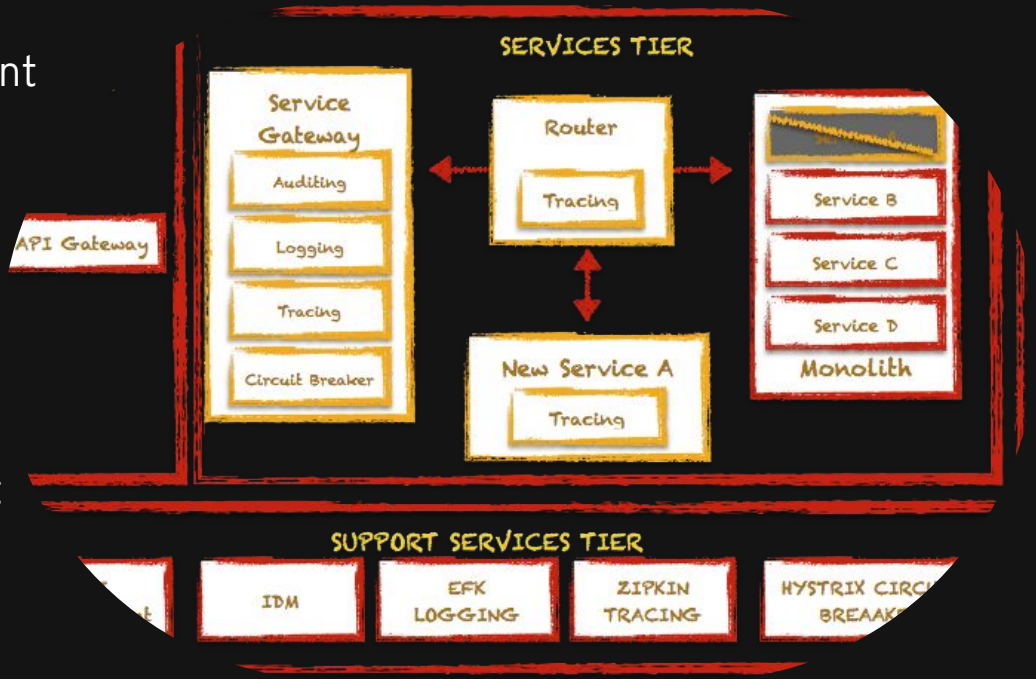
Red Hat

API development

Develop faster

Key Components to API Development

- Lightweight and Modular
- Wide selection of tooling
- Consistent and Repeatable Development Patterns
- API Self Documenting Support

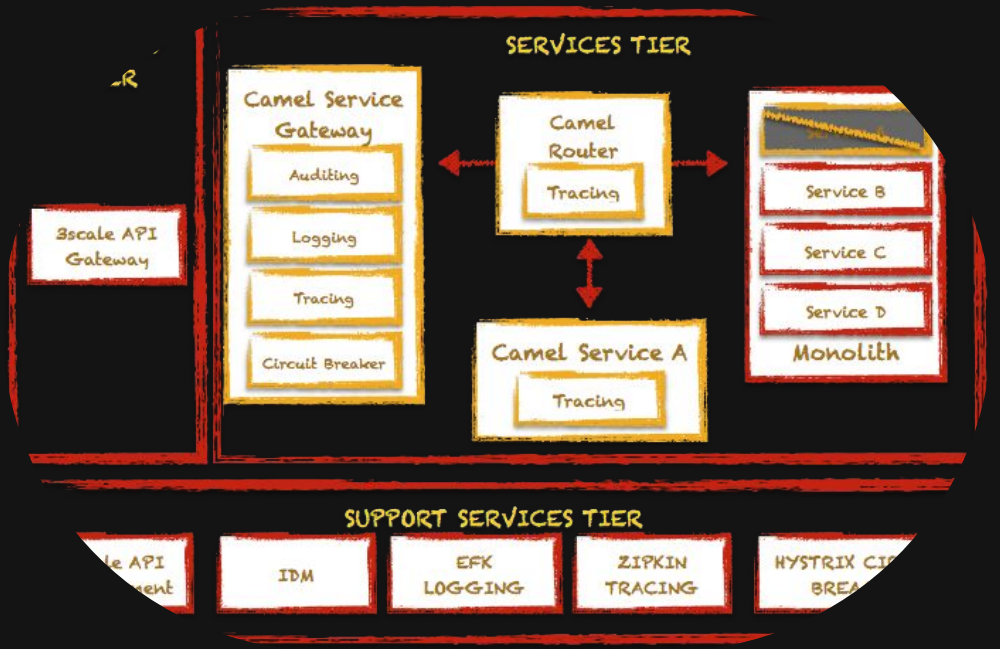


API development

Develop APIs faster with Red Hat Fuse

Why JBoss Fuse

- Modular AppDev framework
- Reusable MSA patterns & components
- Resilient error handling and connection management
- Extensive Unit and Integration Testing Framework for Container Pipelines

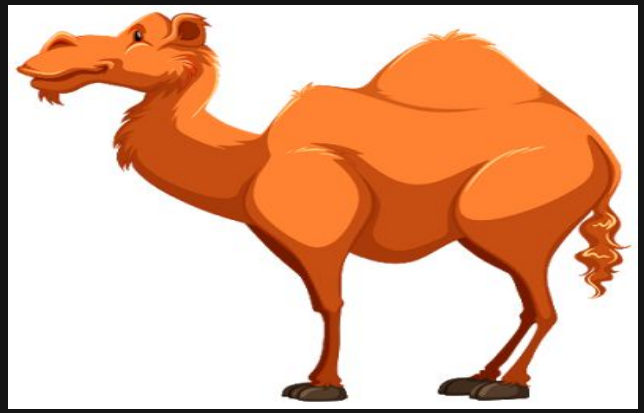


Red Hat

MSA development

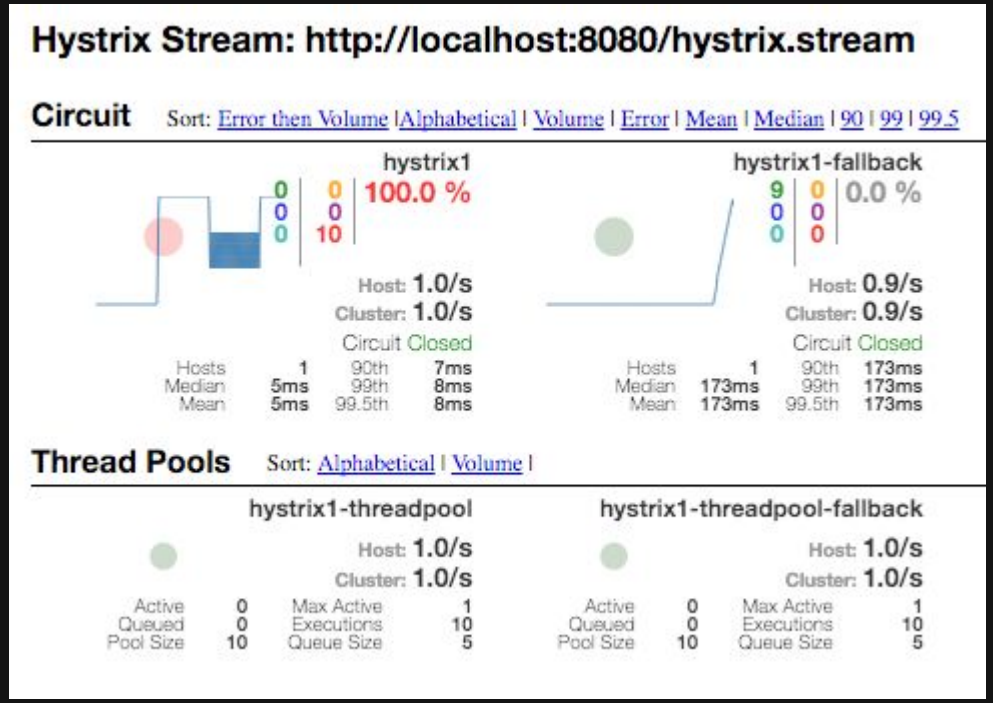
Red Hat Fuse drill down: Apache Camel

- Small Java library
- 200+ components for integrating systems (bring along only the ones you use)
- Powerful EIPs (routing, transformation, error handling)
- Distributed-systems swiss-army knife!
- Declarative DSL
- Embeddable into any JVM (EAP, Karaf, Tomcat, Spring Boot, Dropwizard, Wildfly Swarm, no container, etc)



Apache Camel Hystrix

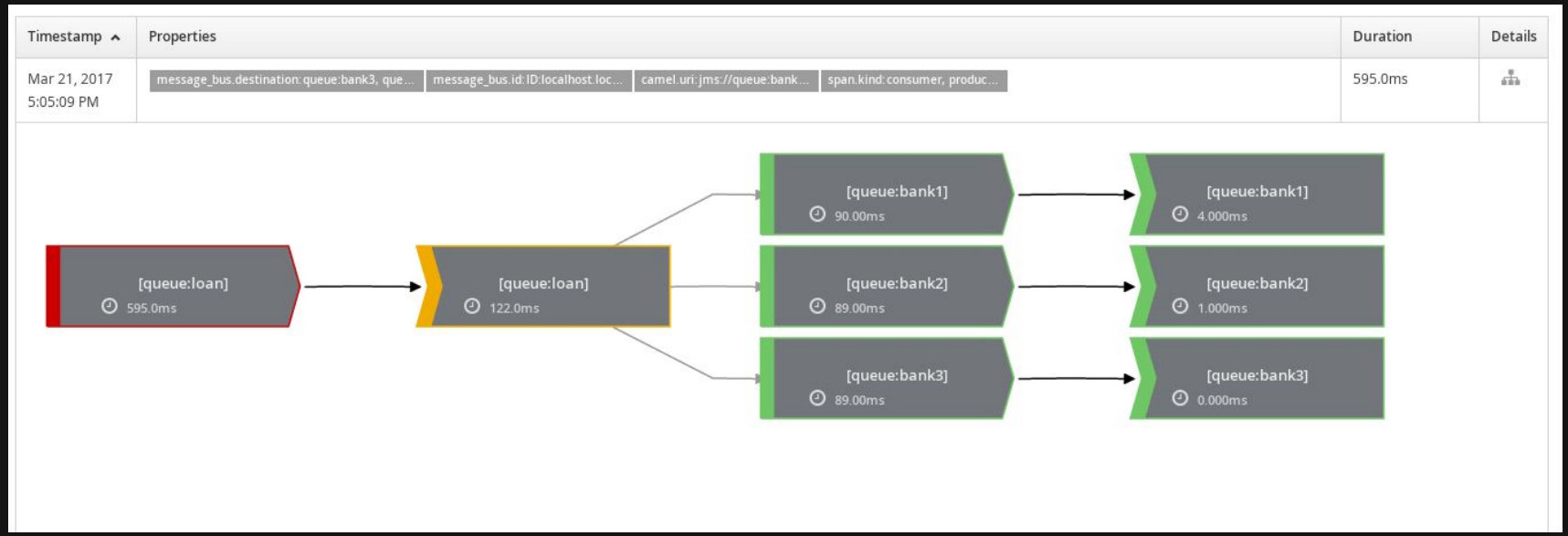
Gain resilience



Red Hat

Camel tracing with Zipkin/Hawkular

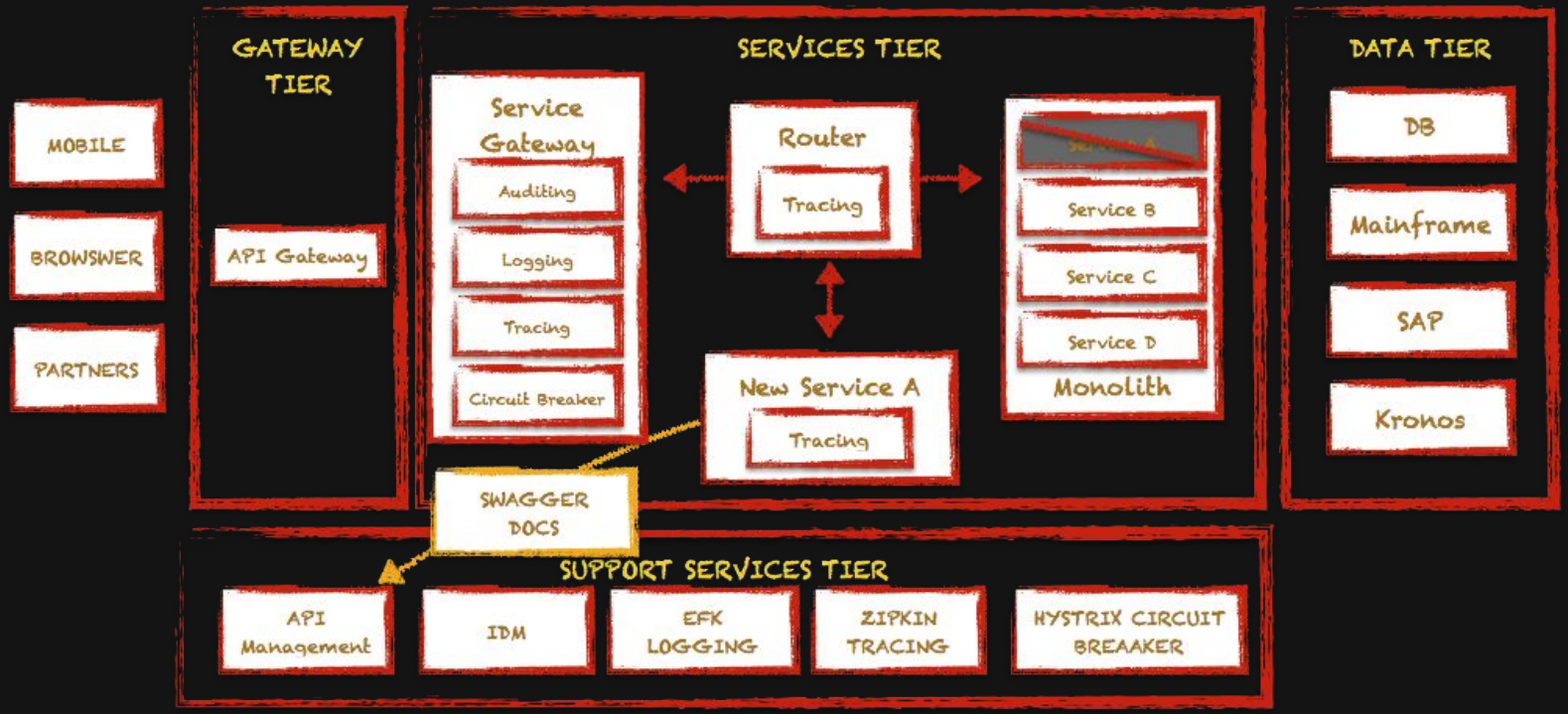
Gain insight



Red Hat

Phase 2: Strangling the monolith

Added bonus!!!



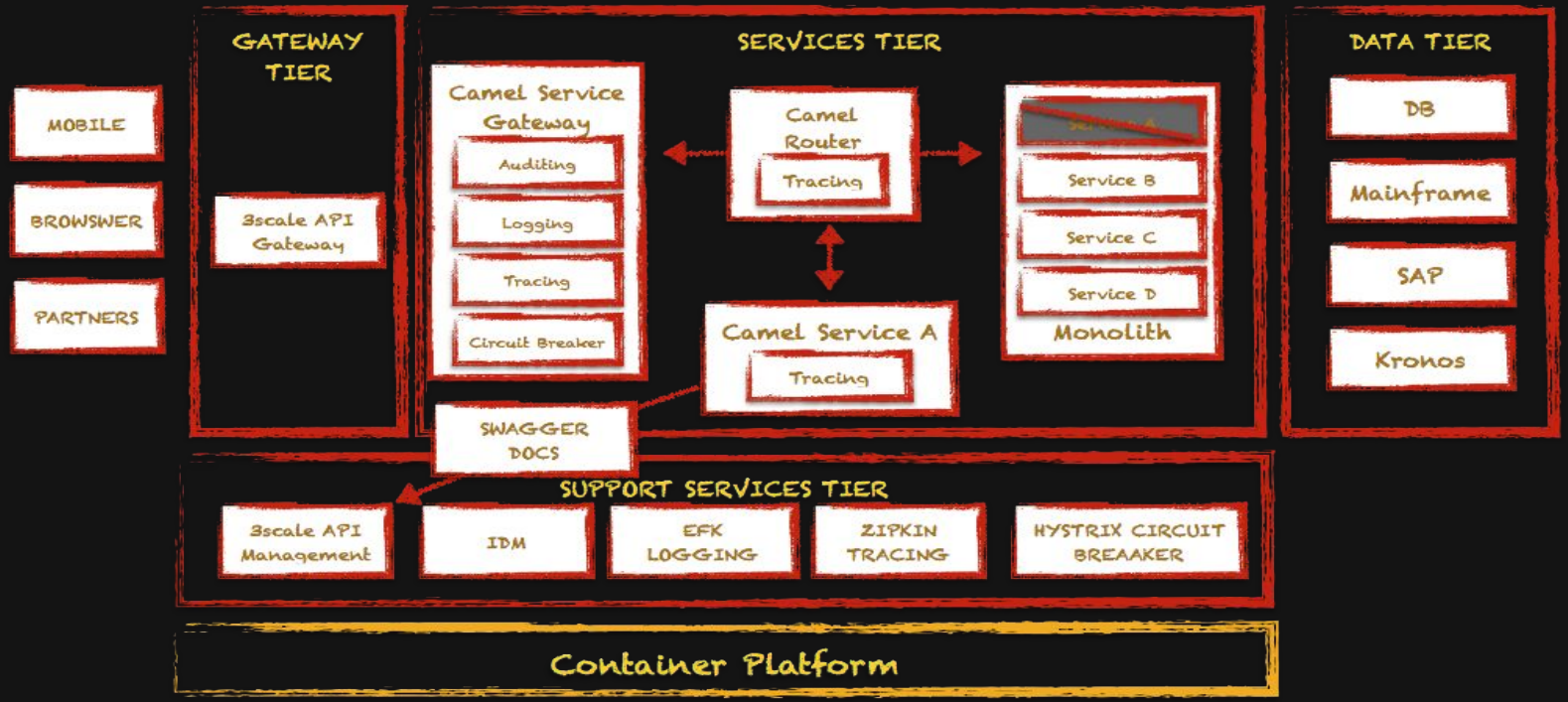
Drill down

Phase 1: Strangling the monolith

Containerization of APIs

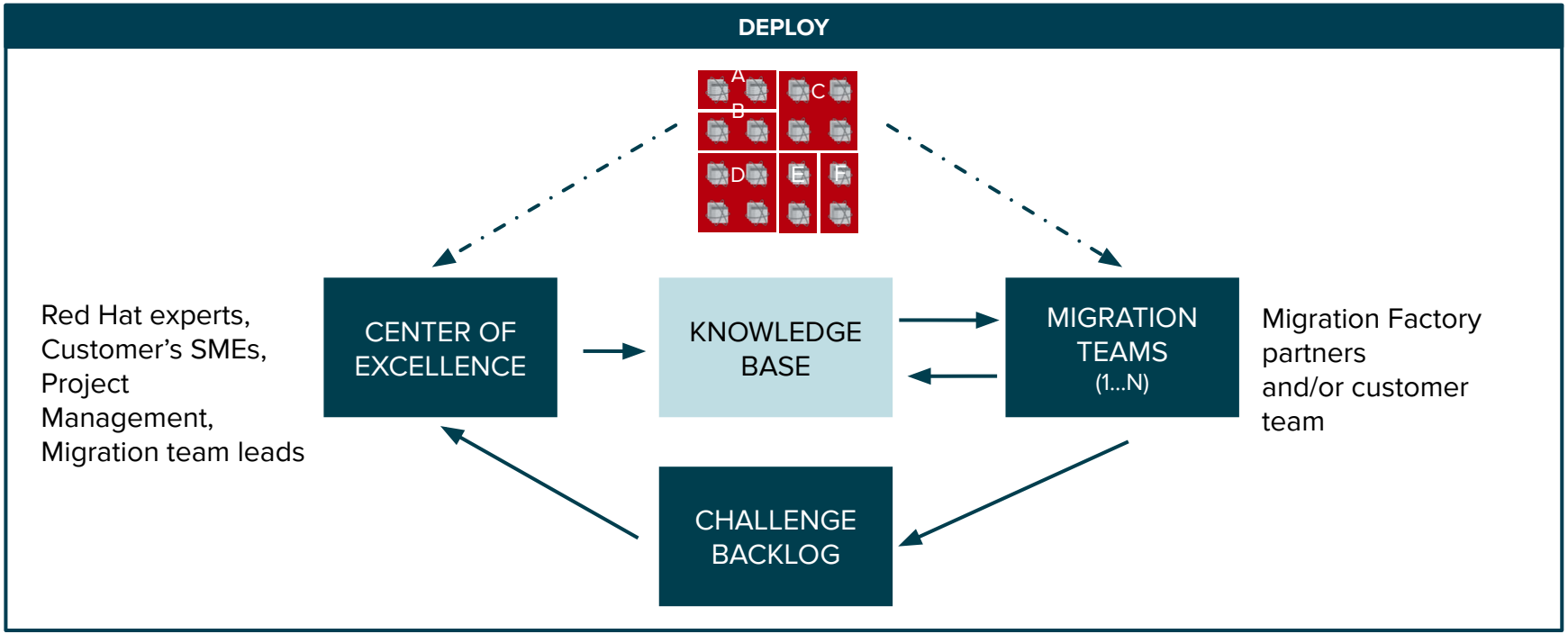
API-led modernization

Containerization



Modernization factory

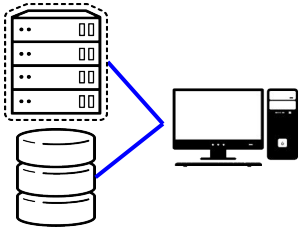
All application types across portfolio



Modern software engineering

Methodical approach to evolve systems

MAINFRAME

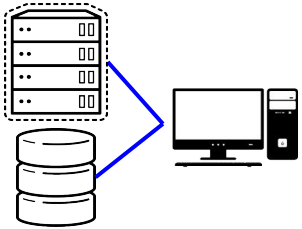


MONOLITH

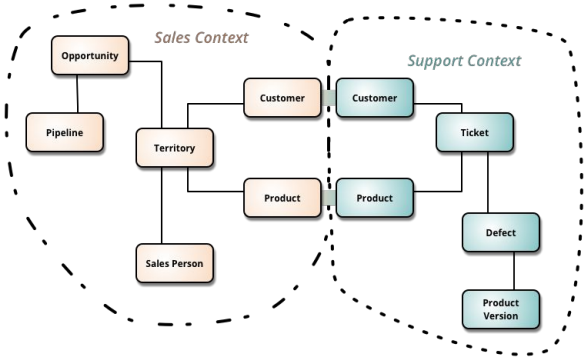
Modern software engineering

Methodical approach to evolve systems

MAINFRAME

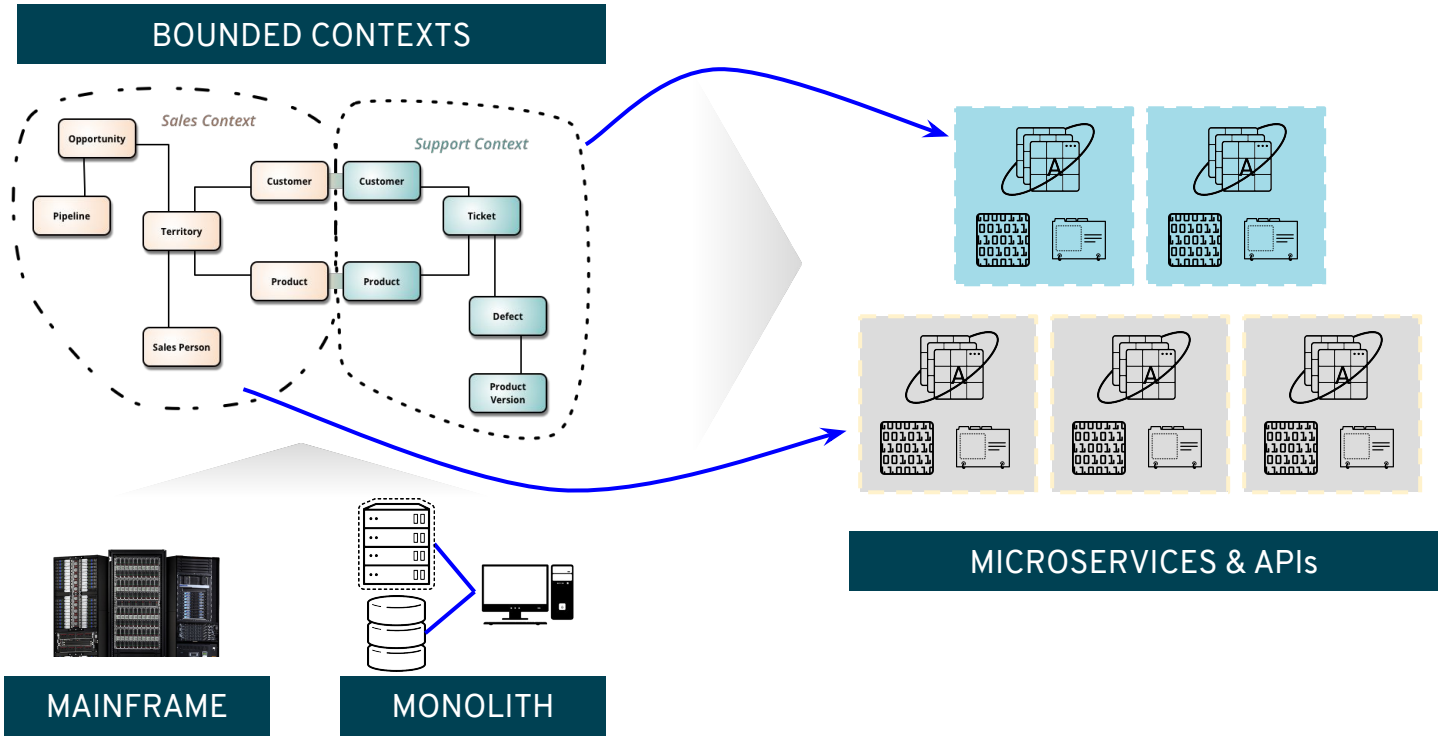


MONOLITH



BOUNDED CONTEXTS

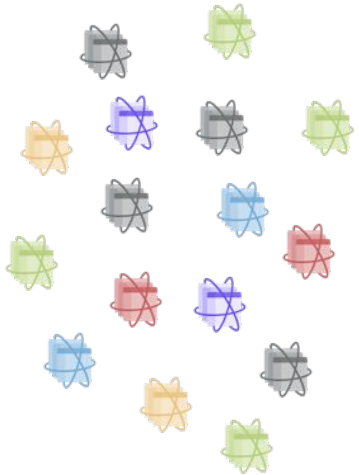
Modern software engineering



Purpose-built tools

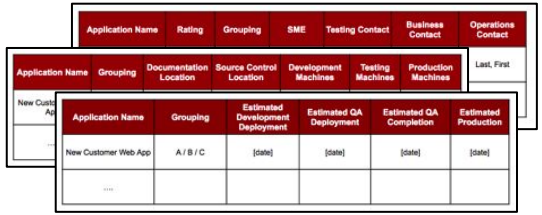
Eliminate guesswork for predictable results

ANALYZE



RATIONALIZE & CATALOG

RED HAT®
APPLICATION
MIGRATION TOOLKIT



PLAN