redhat.

MONOLITHS TO MICROSERVICES: APP TRANSFORMATION

Hands-on Technical Workshop

Thomas Qvarnström
Sr. Technical Marketing Manager
Middleware BU

James Falkner
Sr. Technical Marketing Manager
Middleware BU

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:

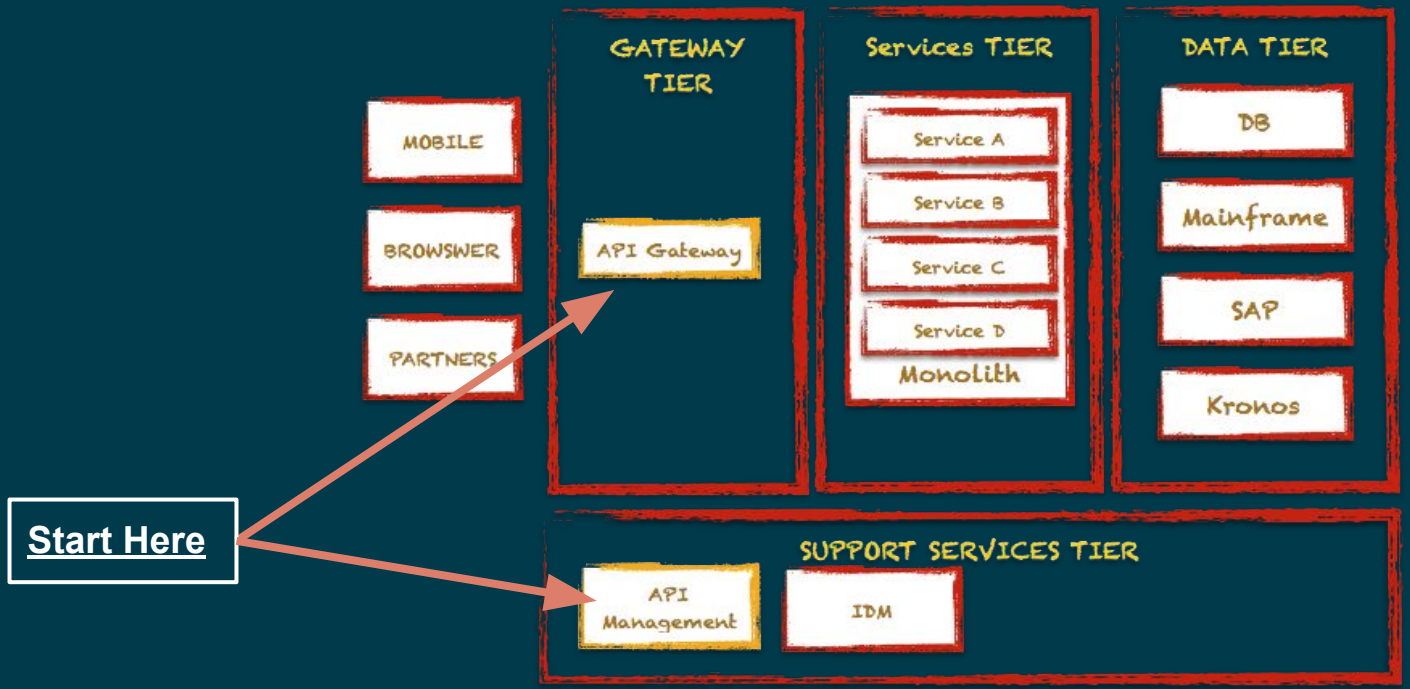
CONNECT AND EXTEND

AKA:

STRANGLING THE MONOLITH

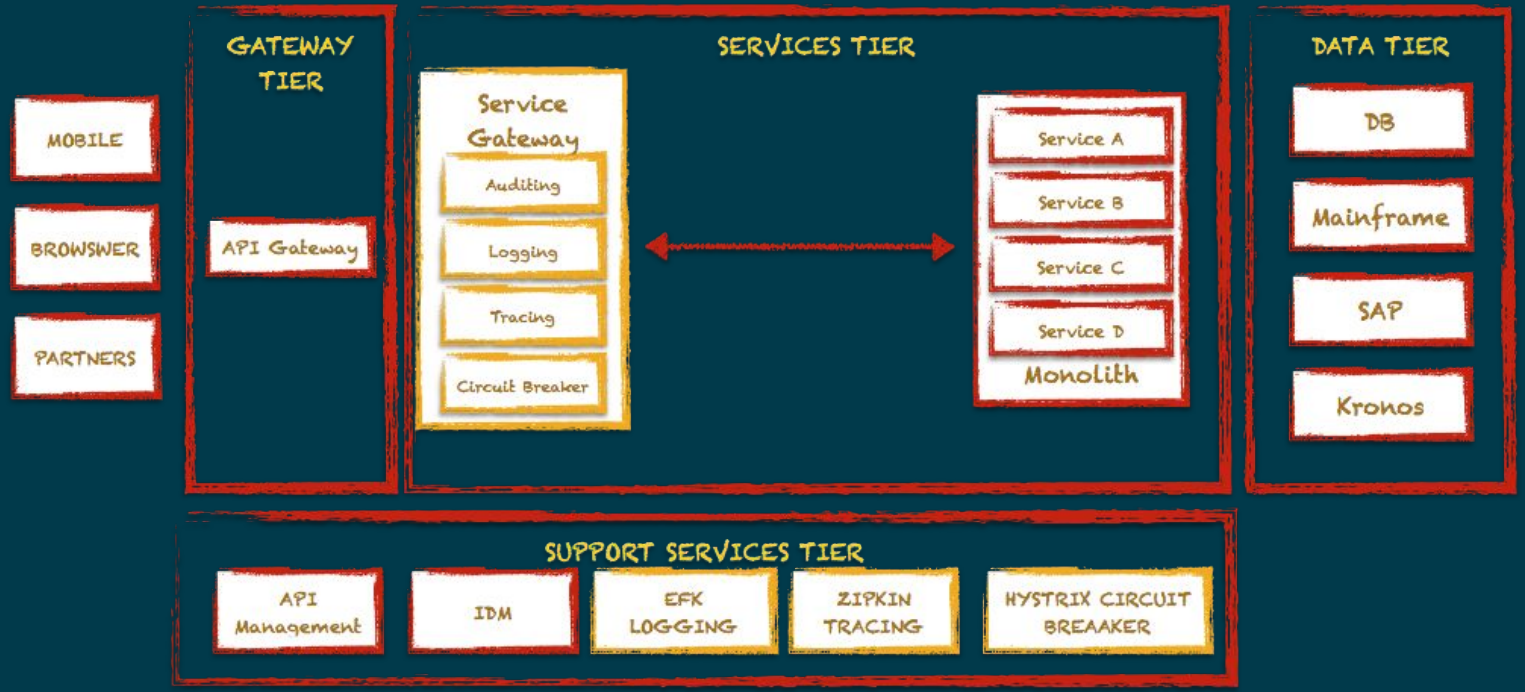
PHASE 1: STRANGLING THE MONOLITH

API First



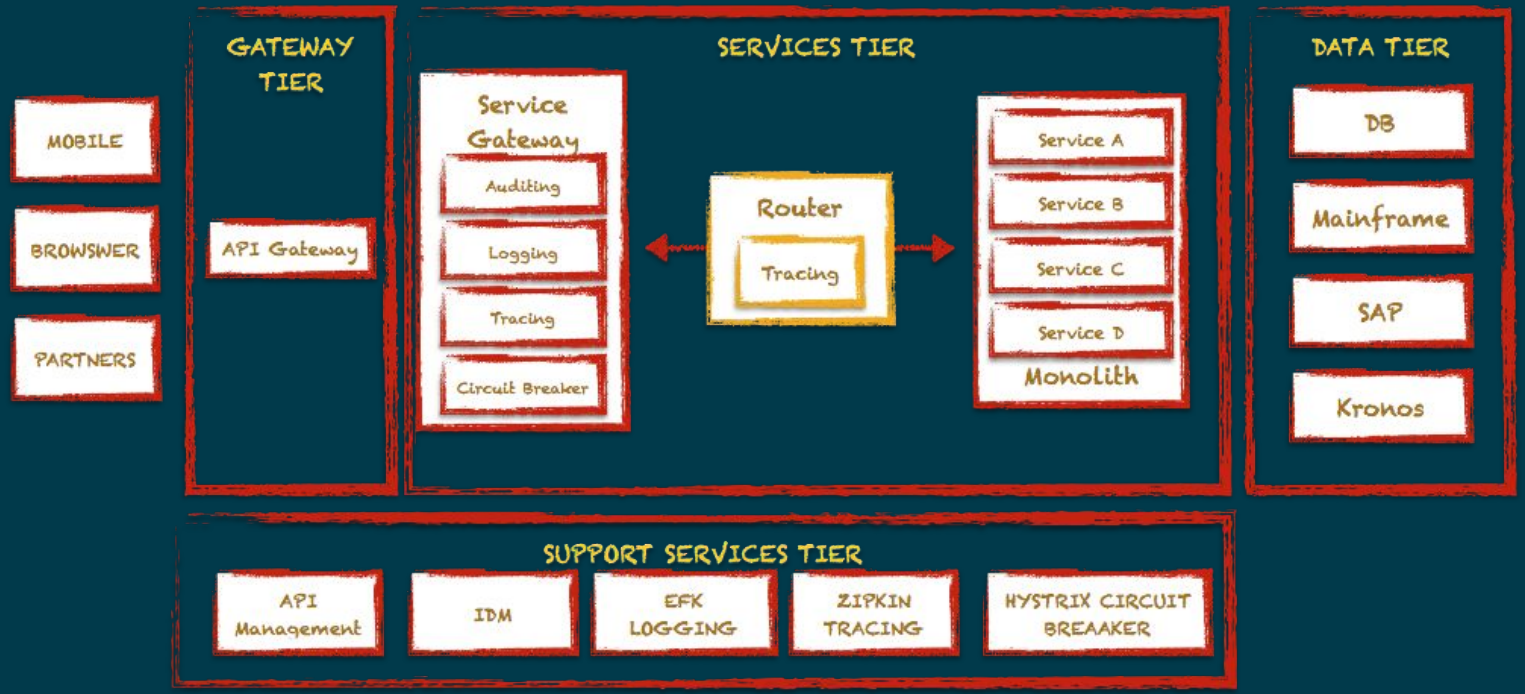
PHASE 2: STRANGLING THE MONOLITH

Connect and extend with a Services Gateway



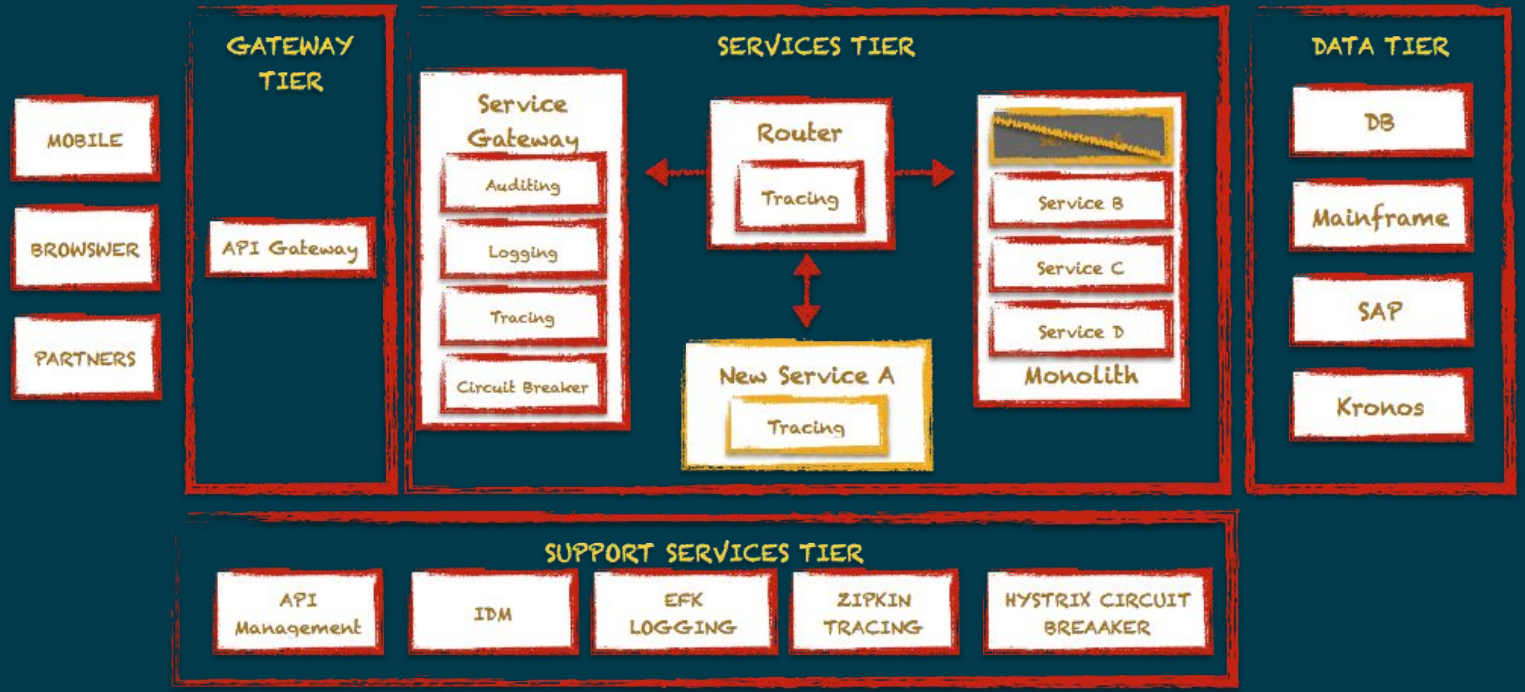
PHASE 2: STRANGLING THE MONOLITH

Create a path for new services



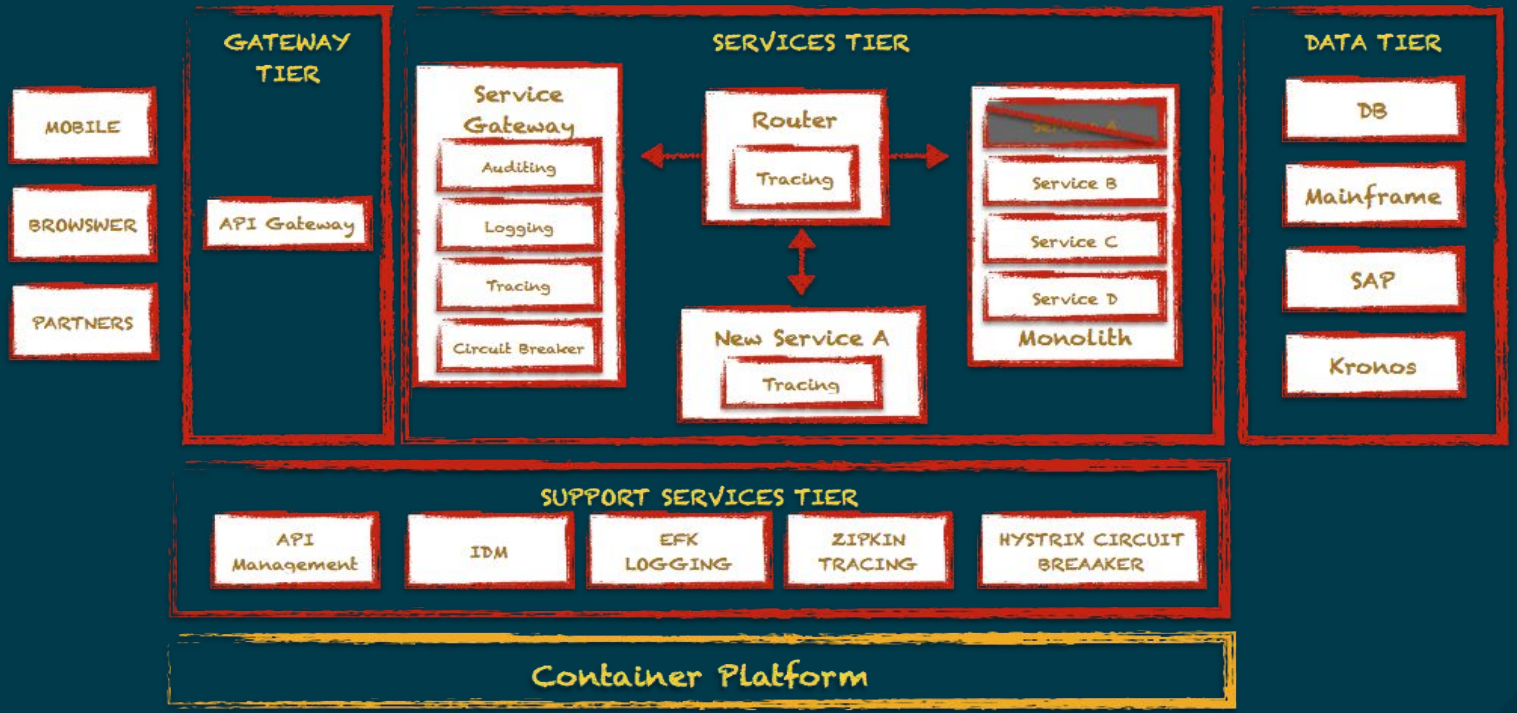
PHASE 2: STRANGLING THE MONOLITH

Out with the old, in with the new



PHASE 3: STRANGLING THE MONOLITH

Containerization



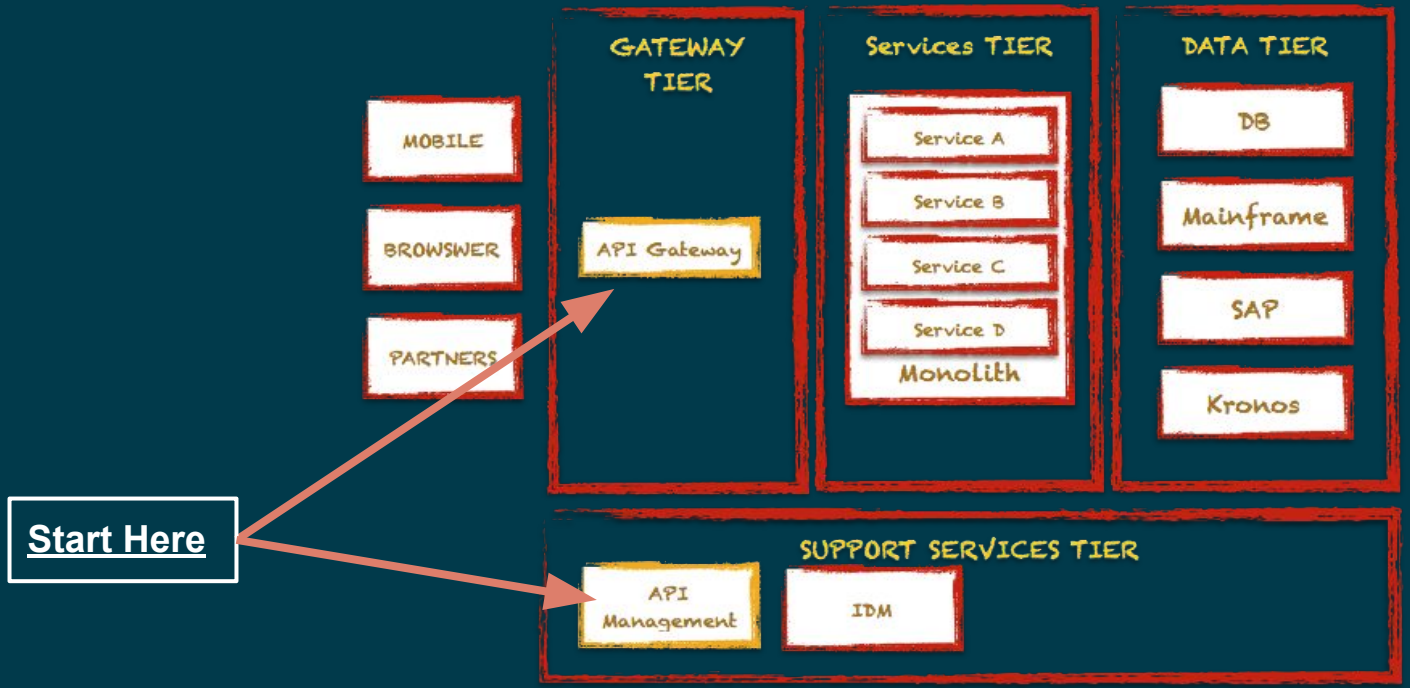
DRILL DOWN

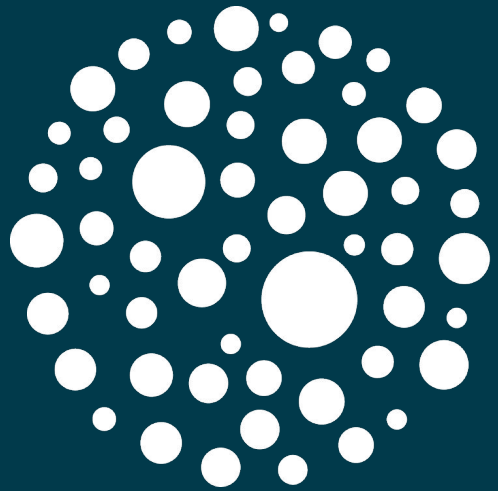
PHASE 1: STRANGLING THE MONOLITH

EXPOSING APIs

PHASE 1: STRANGLING THE MONOLITH

Expose APIs





3scale

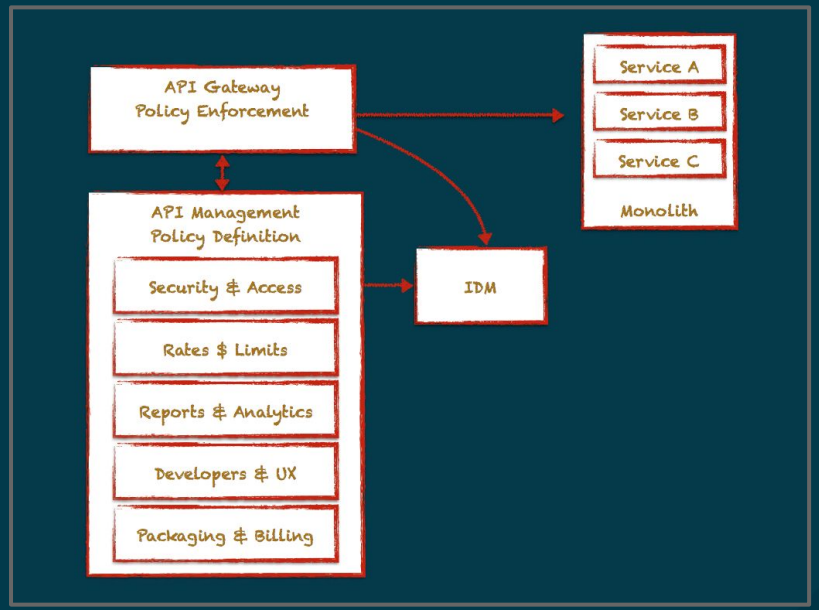
BY RED HAT®

WHY API MANAGEMENT?

API Management is where we start growing and learning

5 Components to API Management

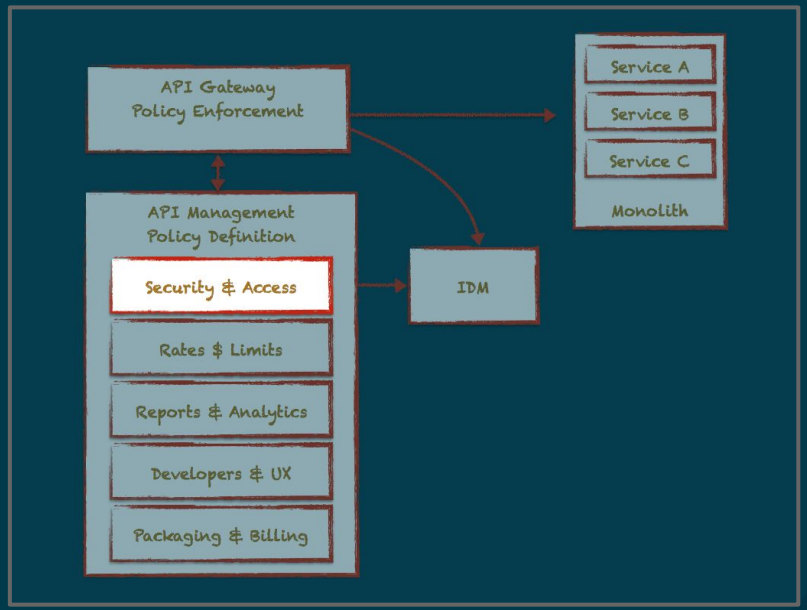
- Security & Access Control
- API Contracts, Throttling & Rate Limits
- Reports & Analytics
- Developer & Partner UX
- Packaging, Billing & Payments



WHY API MANAGEMENT

API Security & Access Control

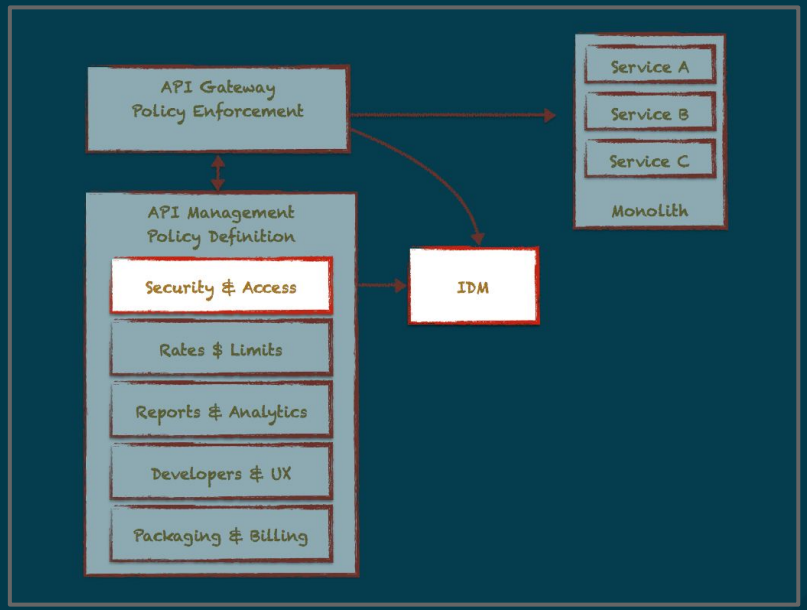
- Authenticate traffic
- Restrict by policy
- Drop unwelcome calls
- Protect backend services
- Generate overage alerts
- Impose rate limits



WHY API MANAGEMENT

API Security & Access Control

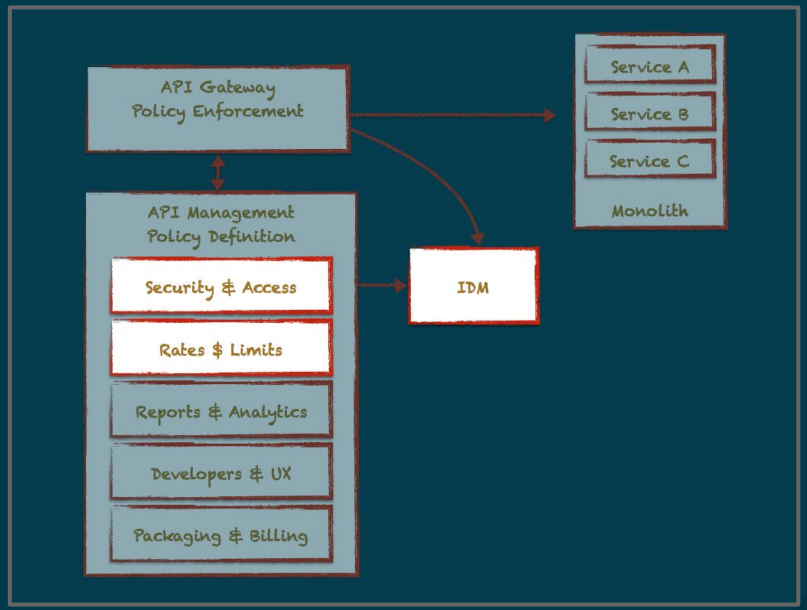
- Multiple authentication mechanisms
  - API Key
  - APP ID / API Key
  - OAuth 2.0



WHY API MANAGEMENT

API Contracts, Throttling & Rate Limits

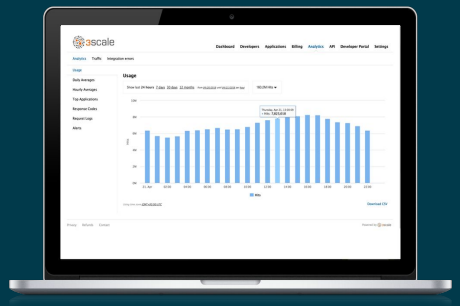
- Allow/restrict access to your API endpoints along with rate limits
- Rate-limit accounts
- User and endpoint level

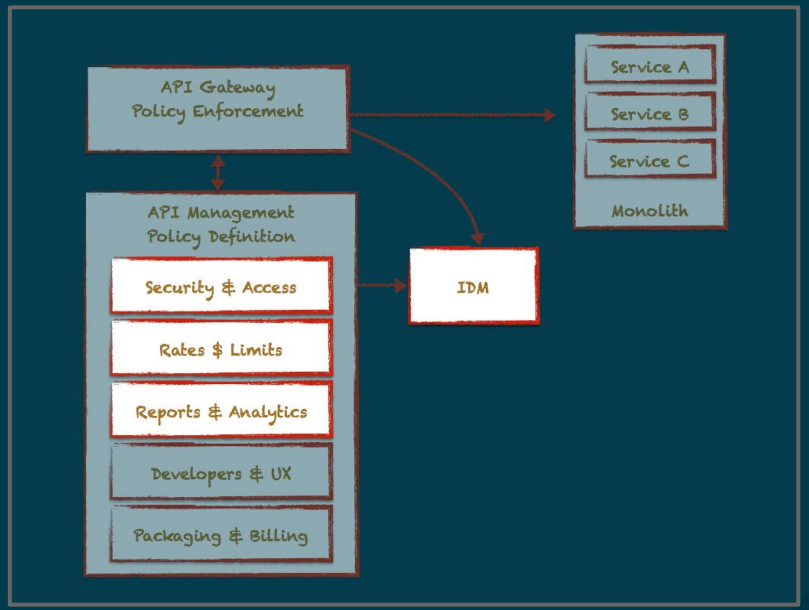


WHY API MANAGEMENT

Reports & Analytics

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

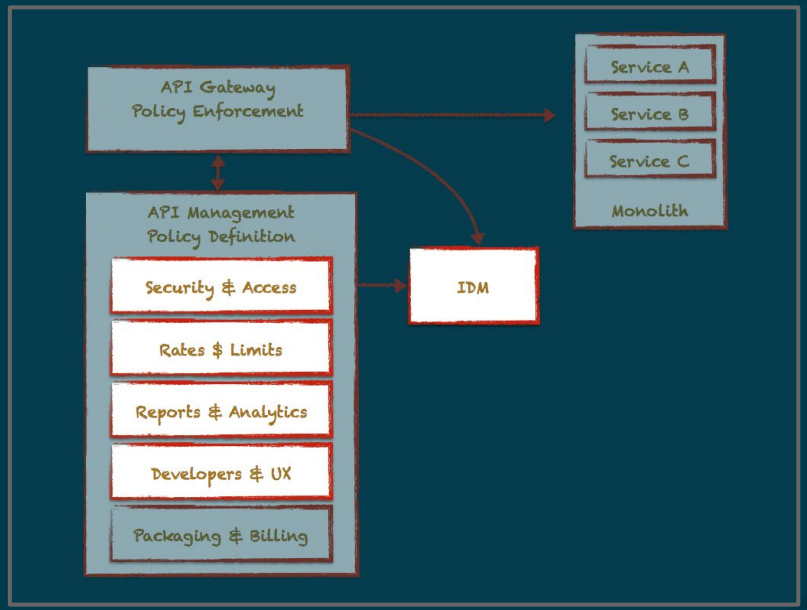




WHY API MANAGEMENT

DEVELOPER PORTAL & USER EXPERIENCE

- Your brand
- Your user experience
- Your user interface

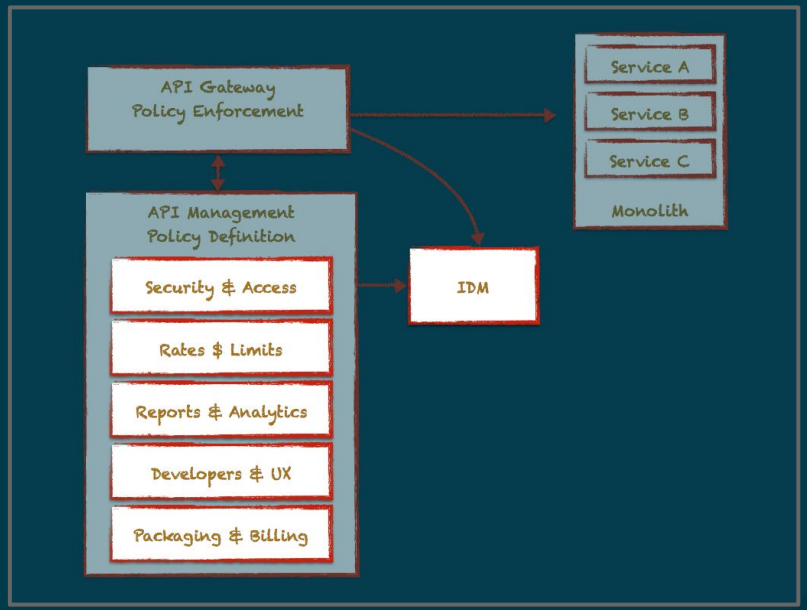


redhat.

WHY API MANAGEMENT

Packaging, Billing & Payments

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

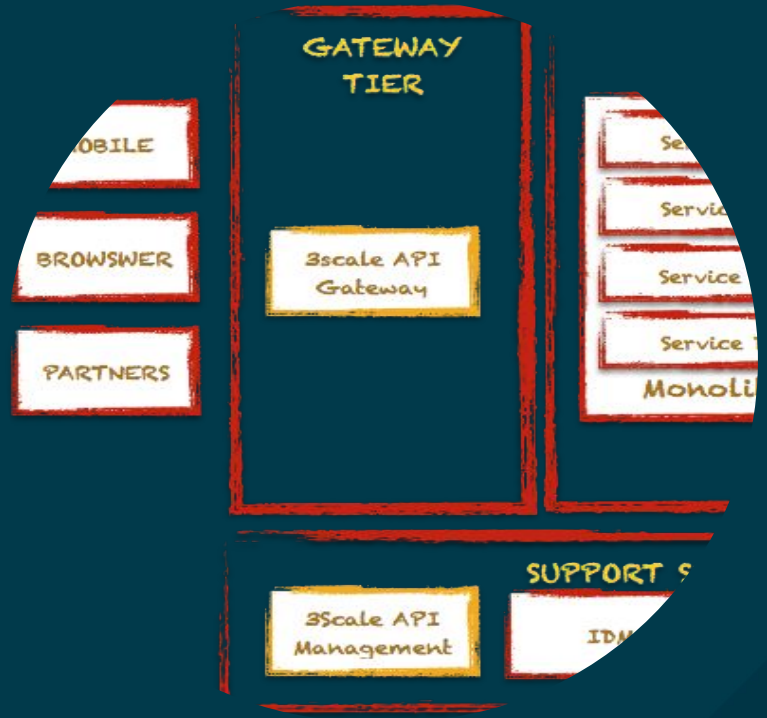


3SCALE API MANAGEMENT

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



redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

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.

RED HAT APP MODERNIZATION PROGRAM

Unified Approach

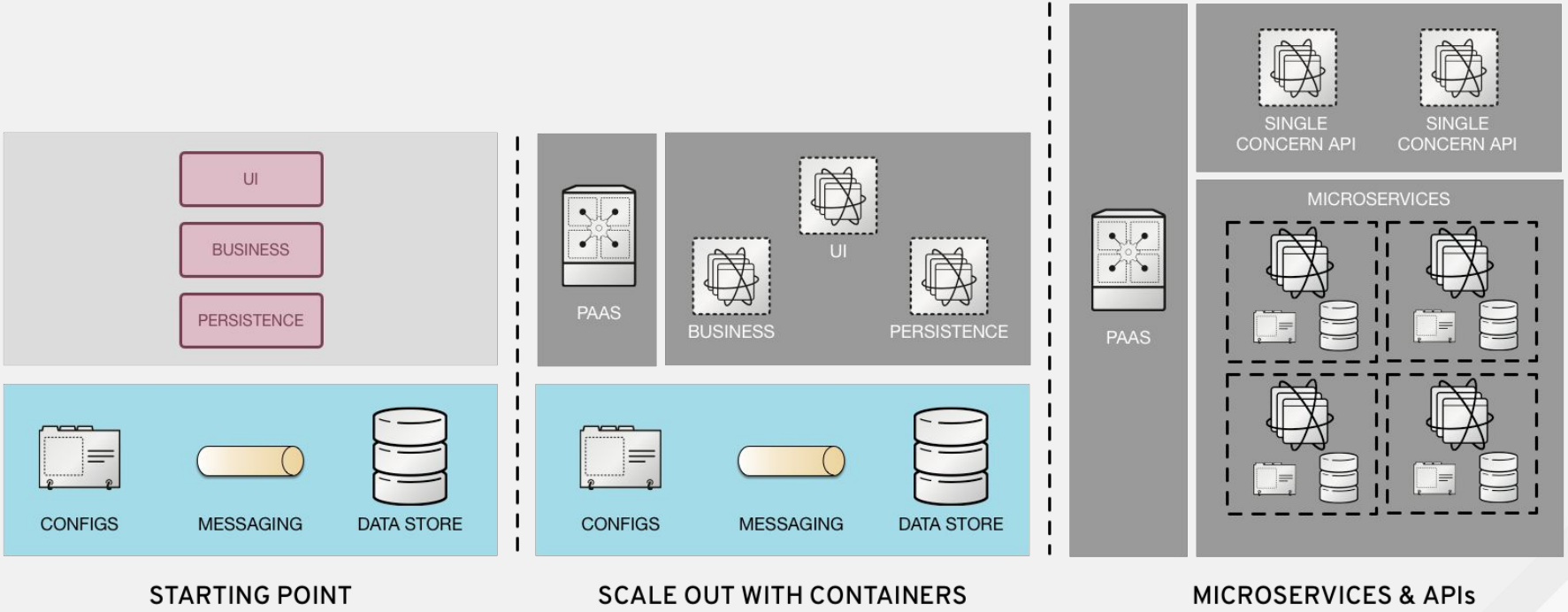
FIELD TESTED PATTERNS
Eliminate technical risk and speedup execution

MODERN SOFTWARE ENGINEERING
Methodical approach to evolve systems

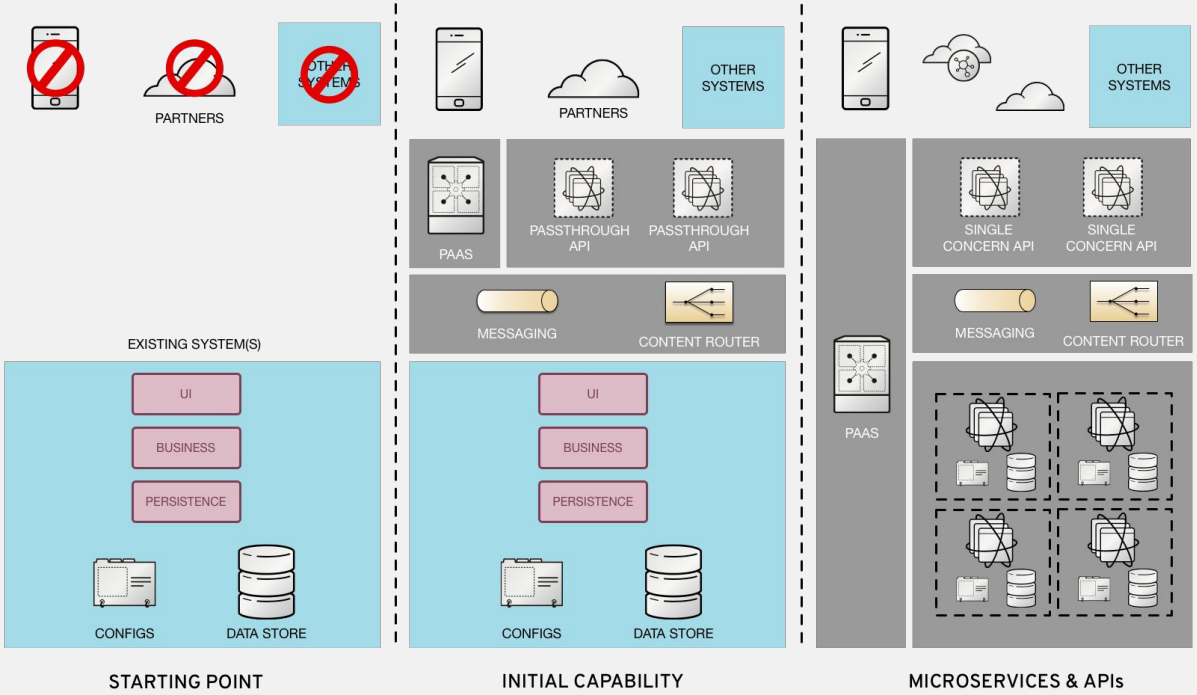
MODERNIZATION FACTORY
Addresses all application types across portfolio

PURPOSE-BUILT TOOLS
Eliminate guesswork for predictable results

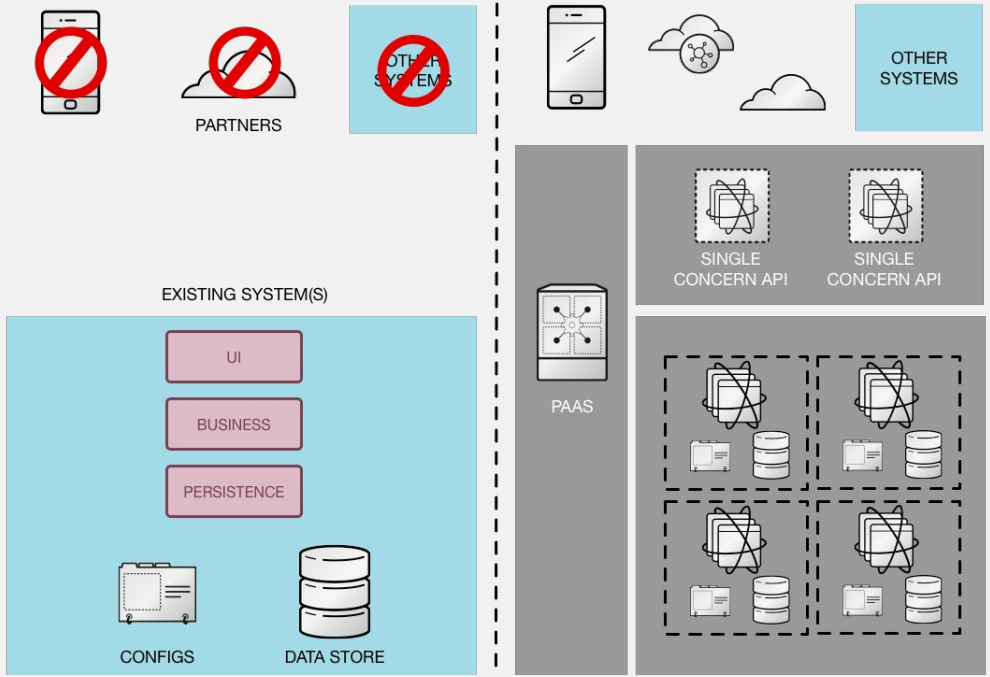
LIFT & SHIFT



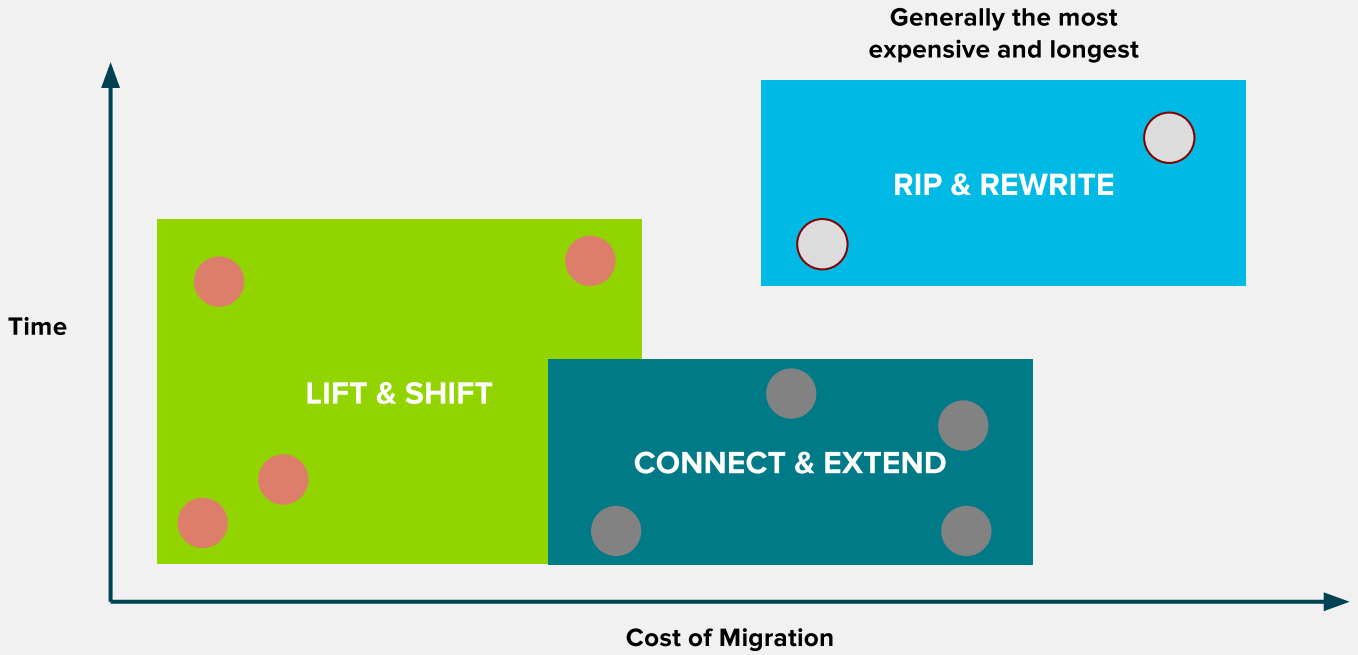
CONNECT & EXTEND



RIP & REWRITE


PARTNERS
OTHER SYSTEMS
EXISTING SYSTEM(S)
UI
BUSINESS
PERSISTENCE
CONFIGS
DATA STORE
OTHER SYSTEMS
SINGLE CONCERN API
SINGLE CONCERN API
PAAS

HOW DO THEY COMPARE?



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

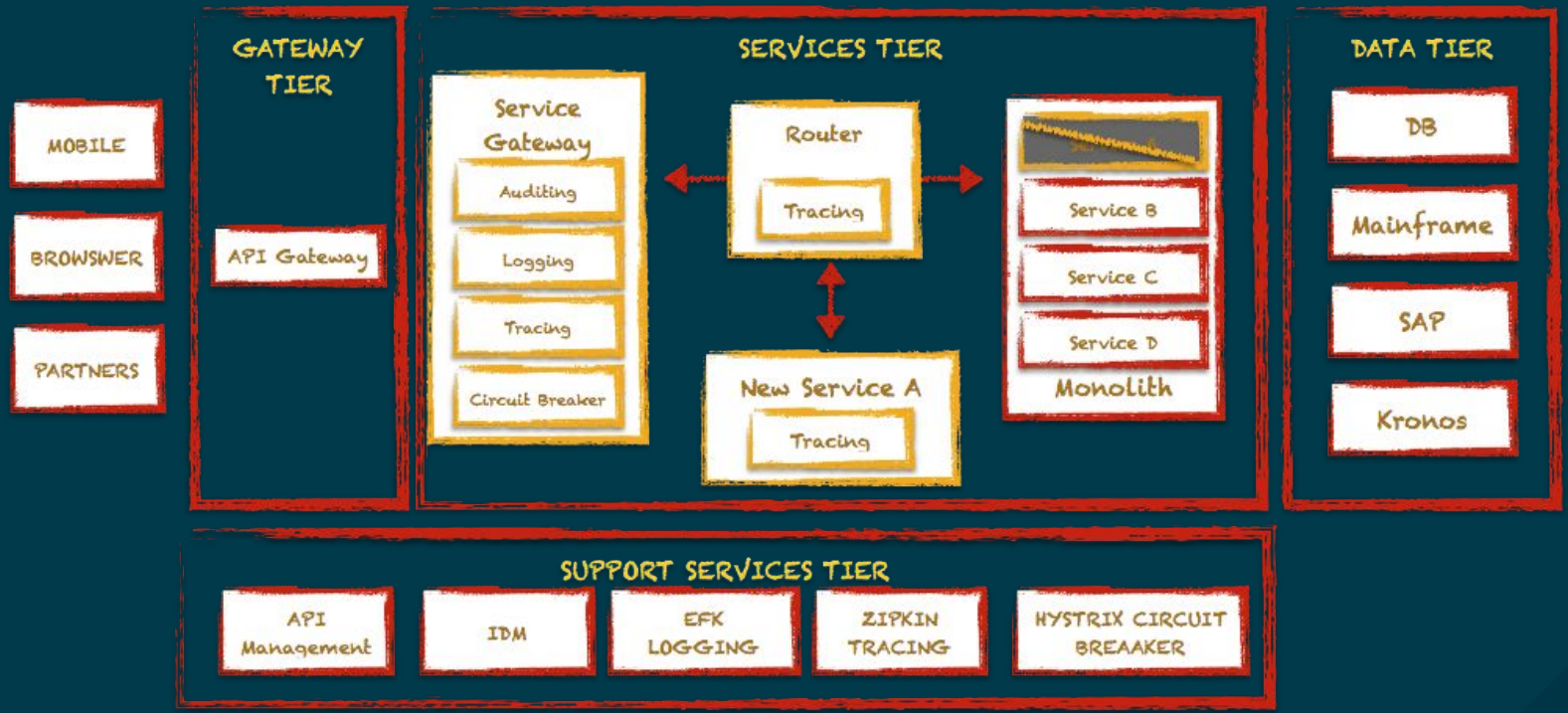
DRILL DOWN

PHASE 2: STRANGLING THE MONOLITH

DEVELOPING APIs

PHASE 2: STRANGLING THE MONOLITH

Developing our new APIs

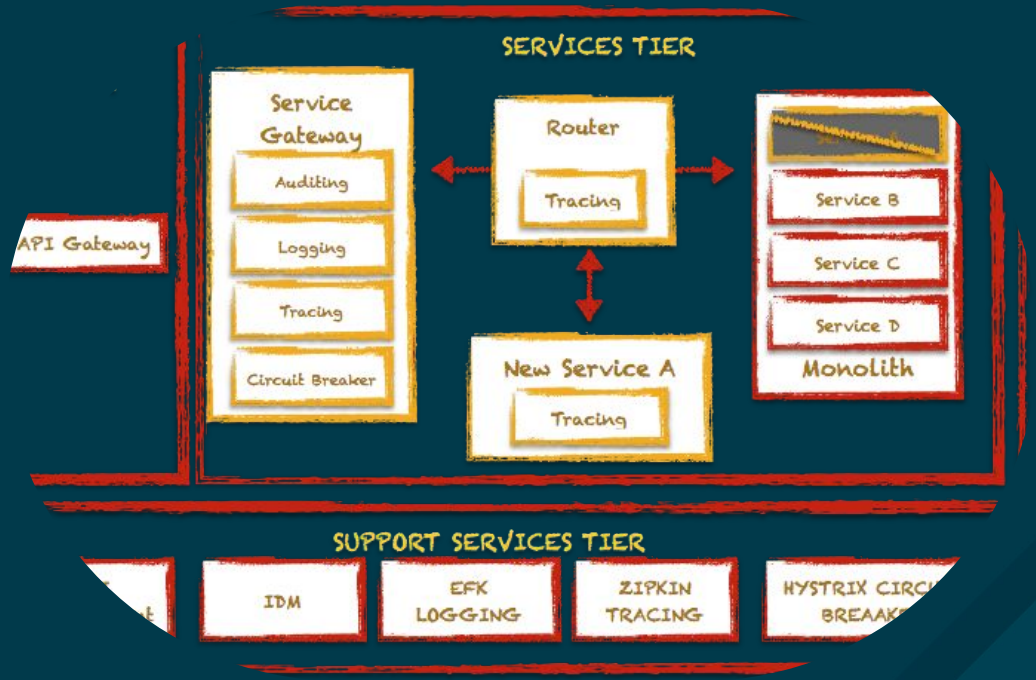


API DEVELOPMENT

Develop Faster

Key Components to API Development

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

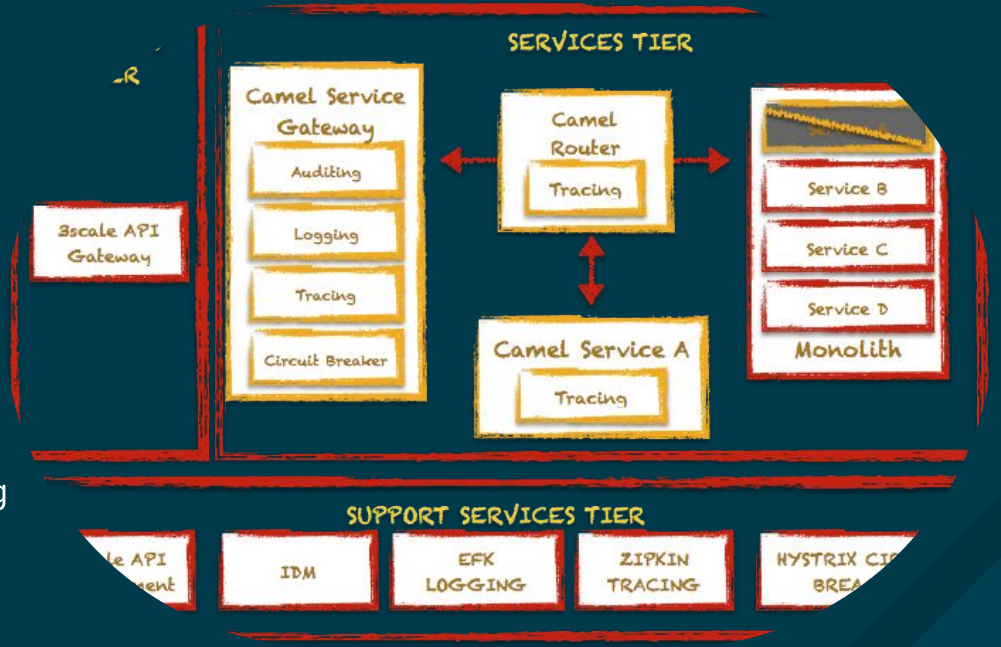


API DEVELOPMENT

Develop APIs Faster with JBoss 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



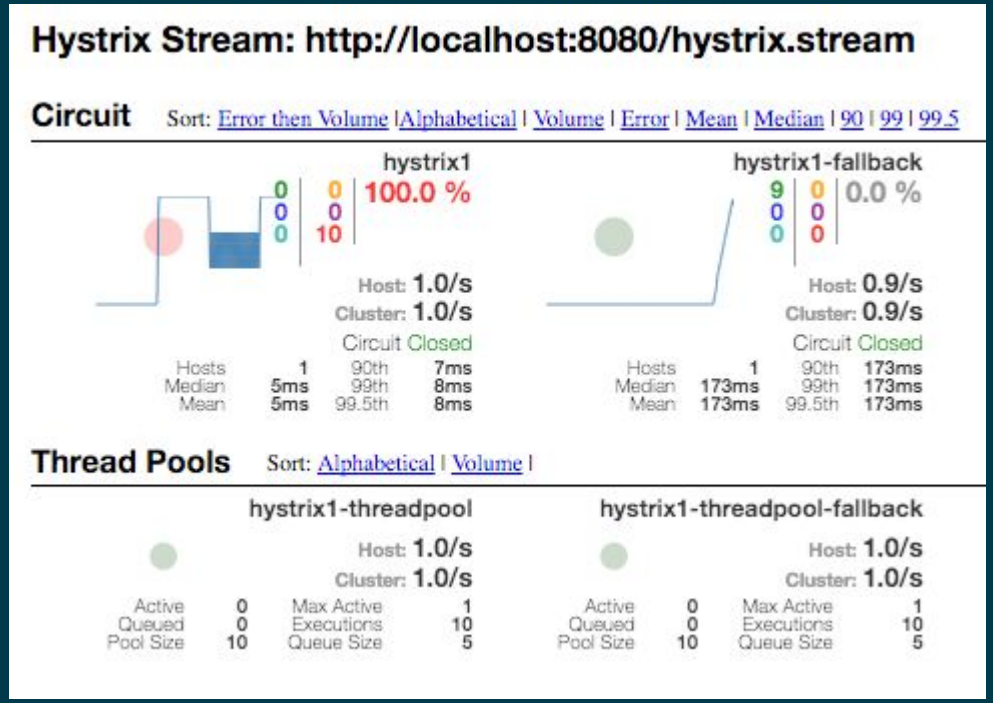
MSA DEVELOPMENT

JBoss 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)

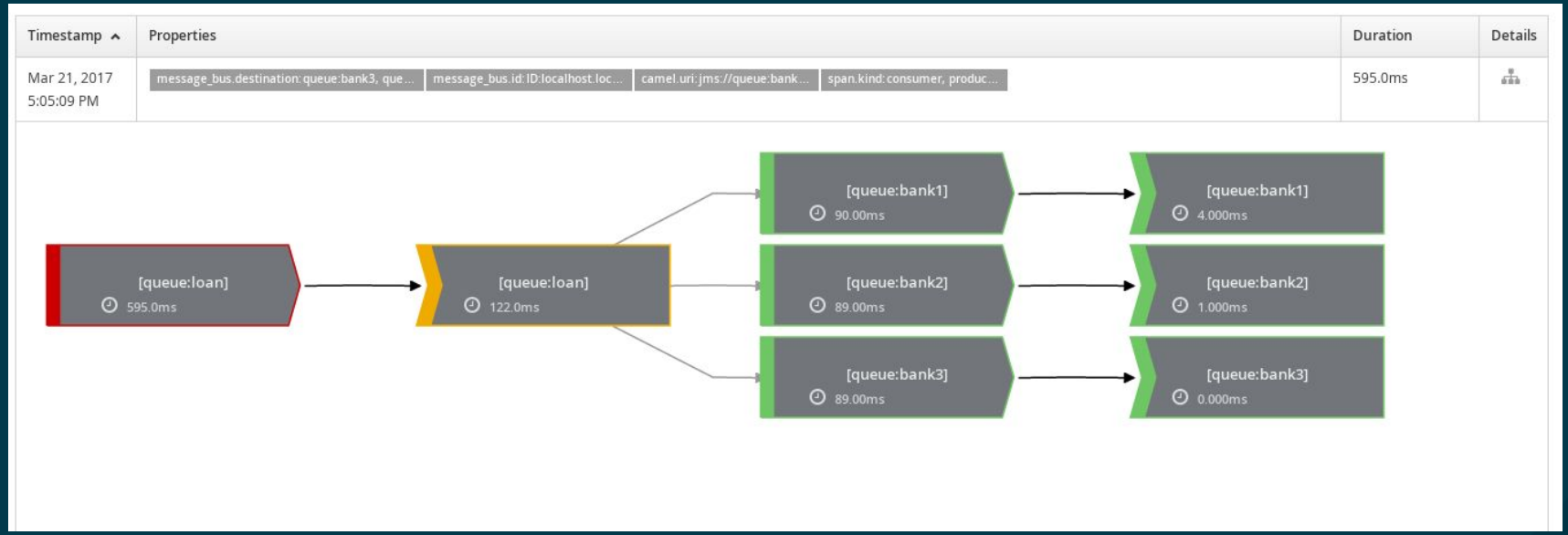
APACHE CAMEL HYSTRIX

GAIN RESILIENCE



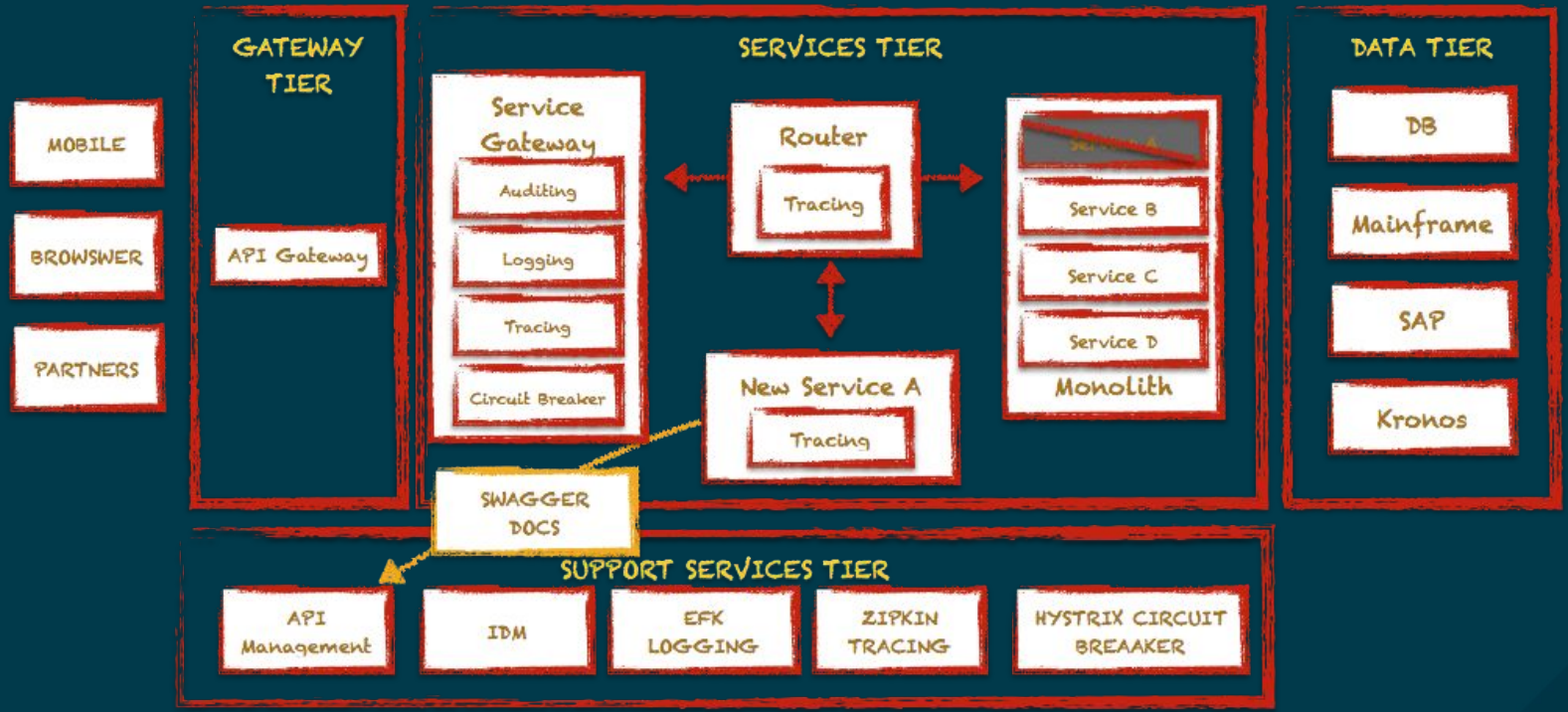
CAMEL TRACING WITH ZIPKIN/HAWKULAR

GAIN INSIGHT



PHASE 2: STRANGLING THE MONOLITH

Added Bonus!!!



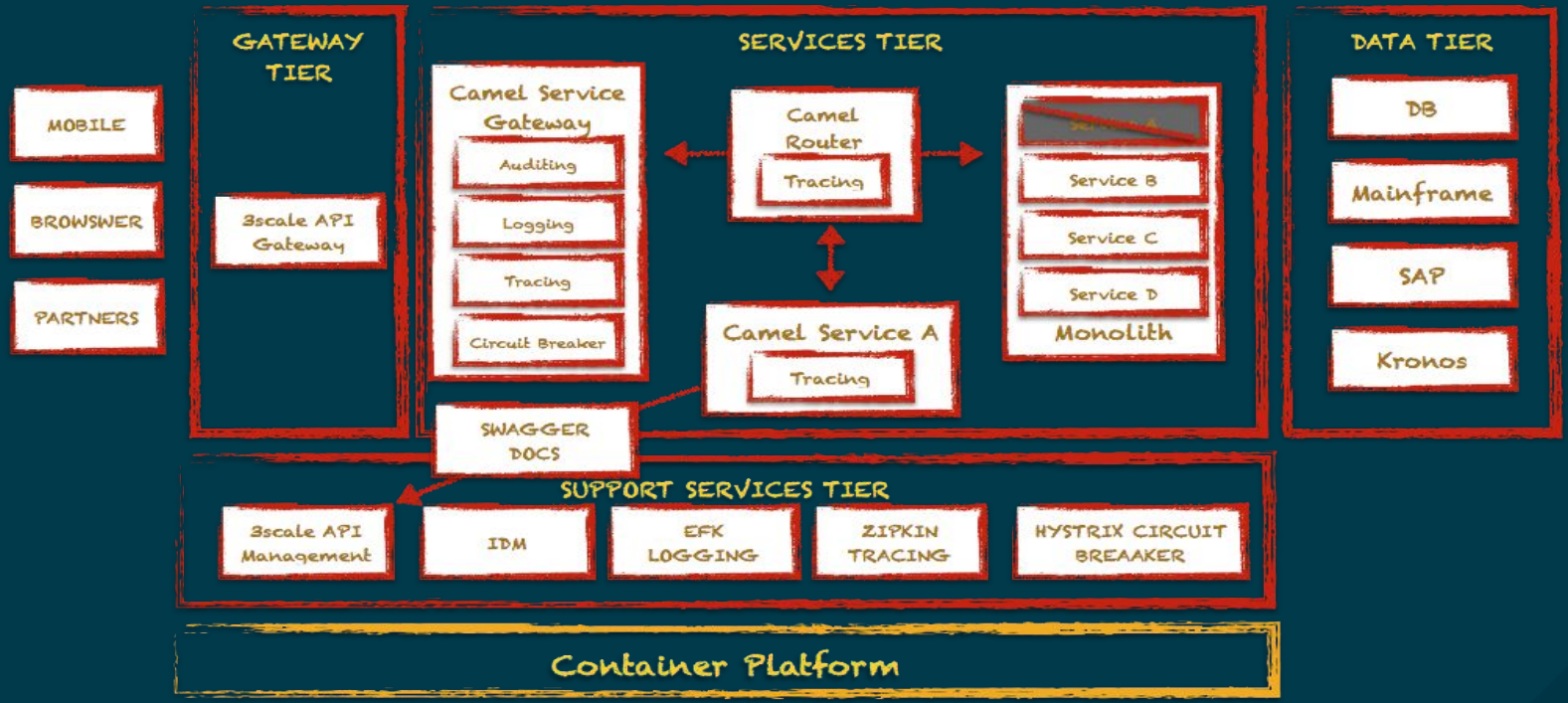
DRILL DOWN

PHASE 3: STRANGLING THE MONOLITH

CONTAINERIZATION OF APIs

API - LED MODERNIZATION

Containerization



RED HAT APP MODERNIZATION PROGRAM

Unified Approach

FIELD TESTED PATTERNS
Eliminate technical risk and speedup execution

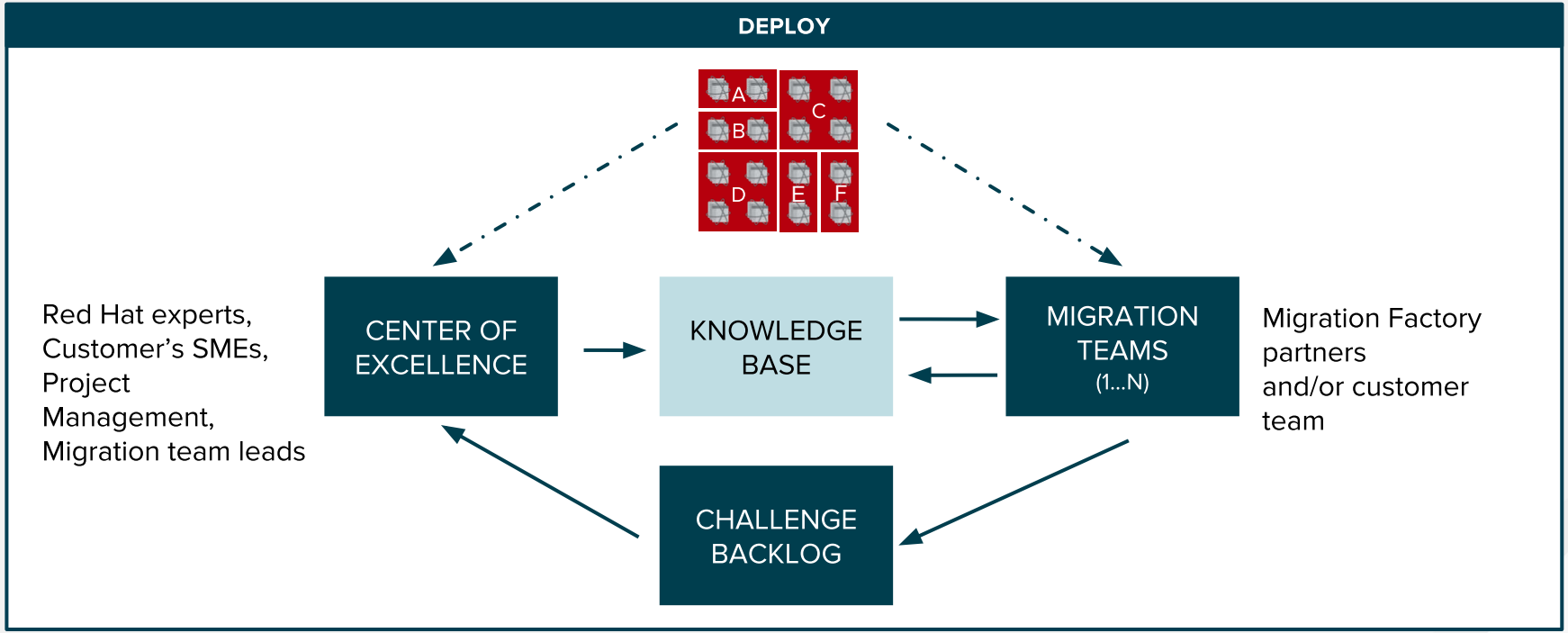
MODERN SOFTWARE ENGINEERING
Methodical approach to evolve systems

MODERNIZATION FACTORY
Addresses all application types across portfolio

PURPOSE-BUILT TOOLS
Eliminate guesswork for predictable results

MODERNIZATION FACTORY

All Application Types across Portfolio



RED HAT APP MODERNIZATION PROGRAM

Unified Approach

FIELD TESTED PATTERNS
Eliminate technical risk and speedup execution

MODERN SOFTWARE ENGINEERING
Methodical approach to evolve systems

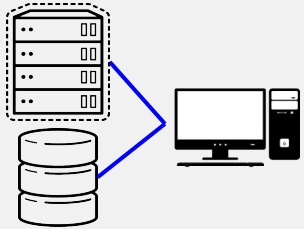
MODERNIZATION FACTORY
Addresses all application types across portfolio

PURPOSE-BUILT TOOLS
Eliminate guesswork for predictable results

MODERN SOFTWARE ENGINEERING

Methodical Approach to Evolve Systems

MAINFRAME

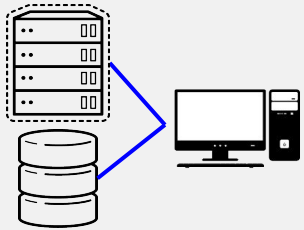


MONOLITH

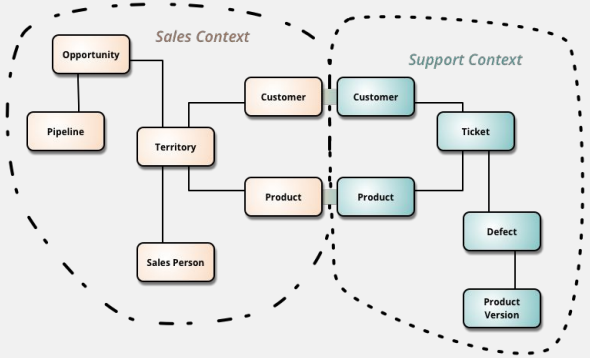
MODERN SOFTWARE ENGINEERING

Methodical Approach to Evolve Systems

MAINFRAME

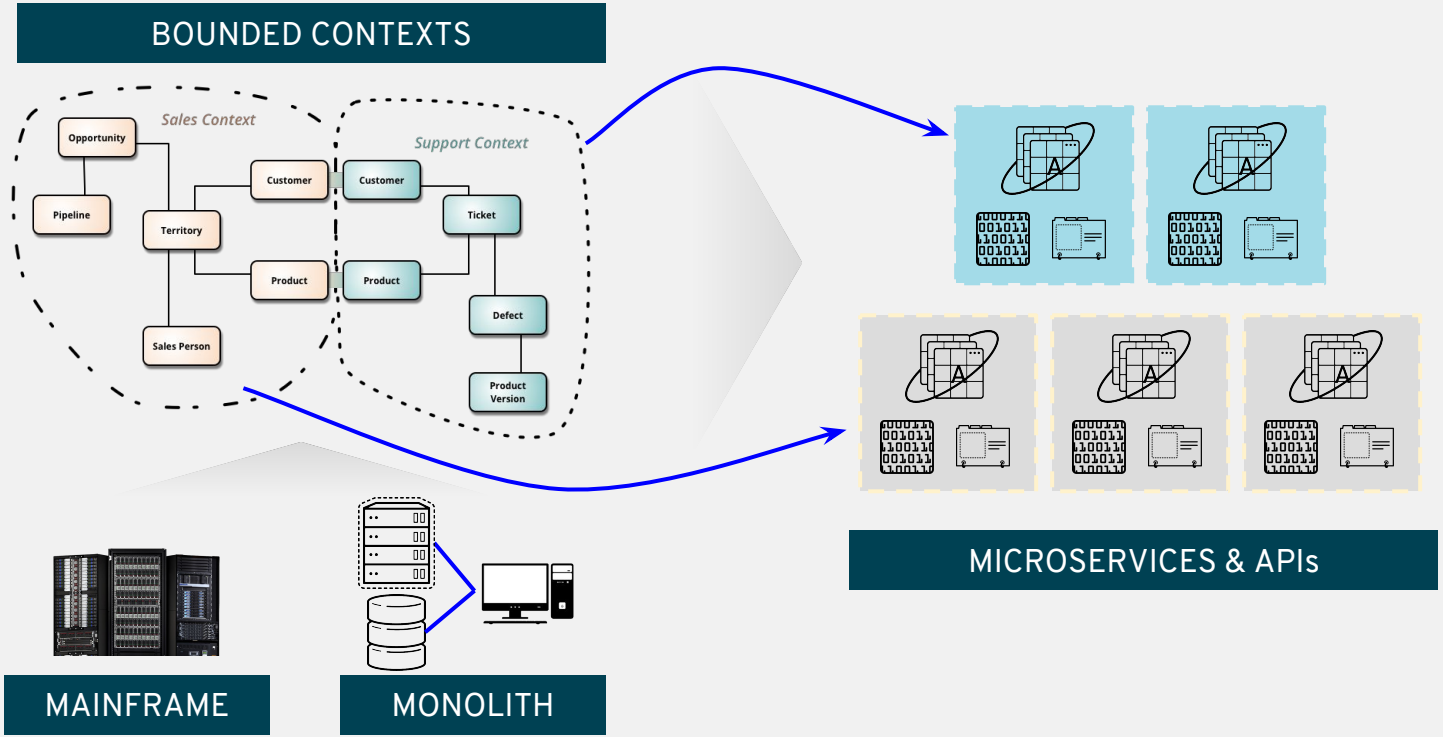


MONOLITH



BOUNDED CONTEXTS

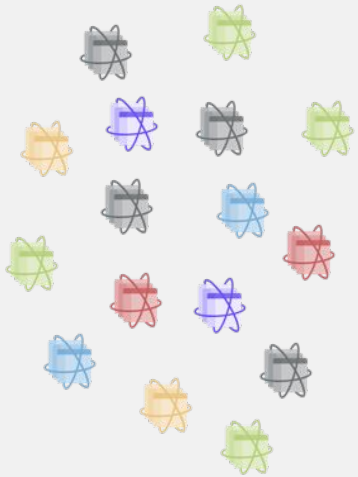
MODERN SOFTWARE ENGINEERING



PURPOSE-BUILT TOOLS

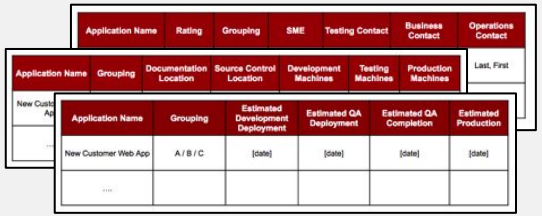
Eliminate guesswork for predictable results

ANALYZE



RATIONALIZE & CATALOG

RED HAT®
APPLICATION
MIGRATION TOOLKIT



PLAN