Engineering the Bank

Designing, delivering and maintaining an Operational Platform for Clydesdale Bank, Yorkshire Bank & B.

Steven O'Day

Technology Consultant Platform Engineering

Phillip Ollenbuttel

Technical Specialist Platform Engineering
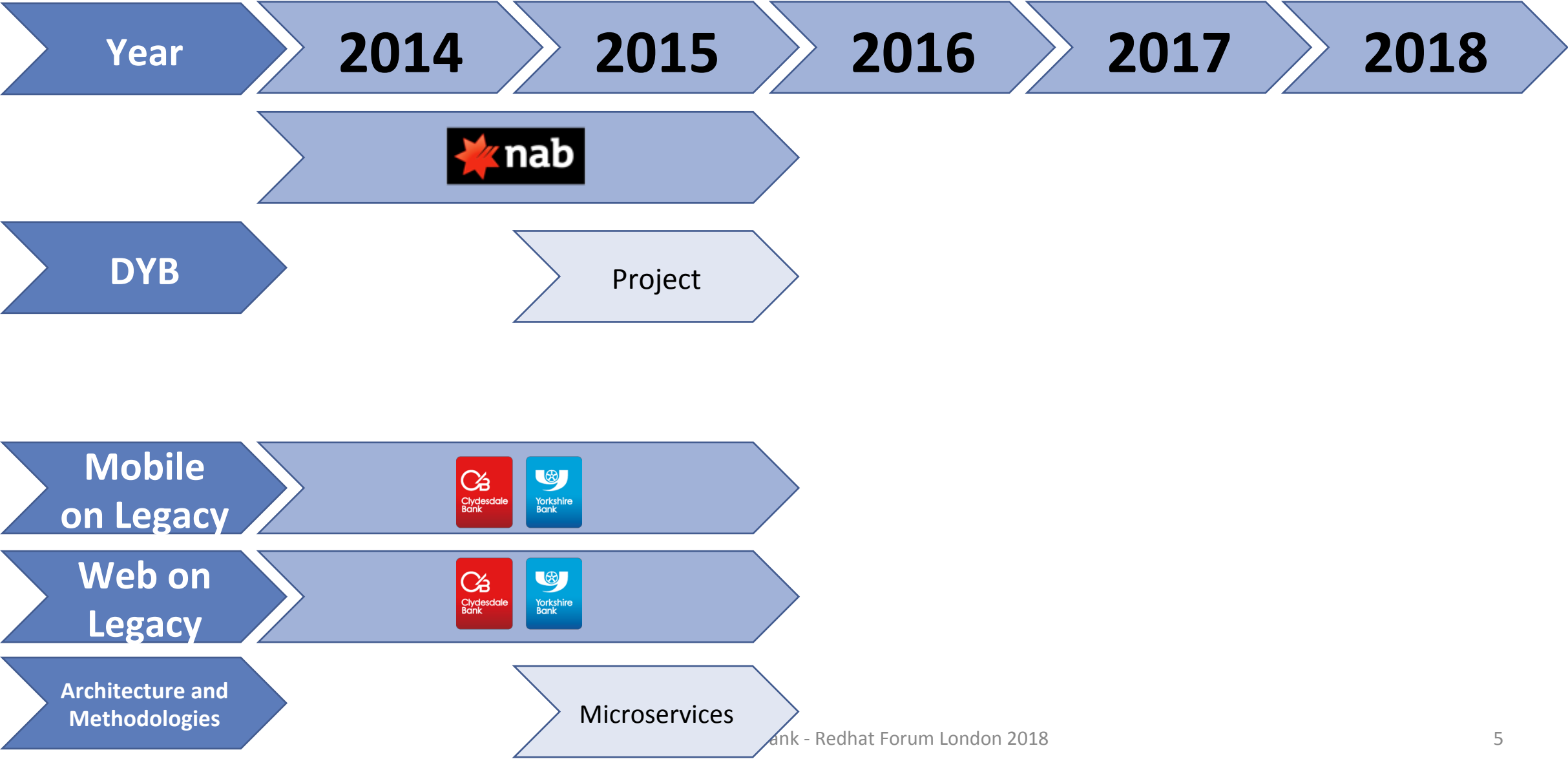
What will we cover?

- Who are CYBG?
- Where we started?
- What did we do?
- What did we learn?
- How did we use those lessons learned?
- Evolution of the Platform Engineering Team
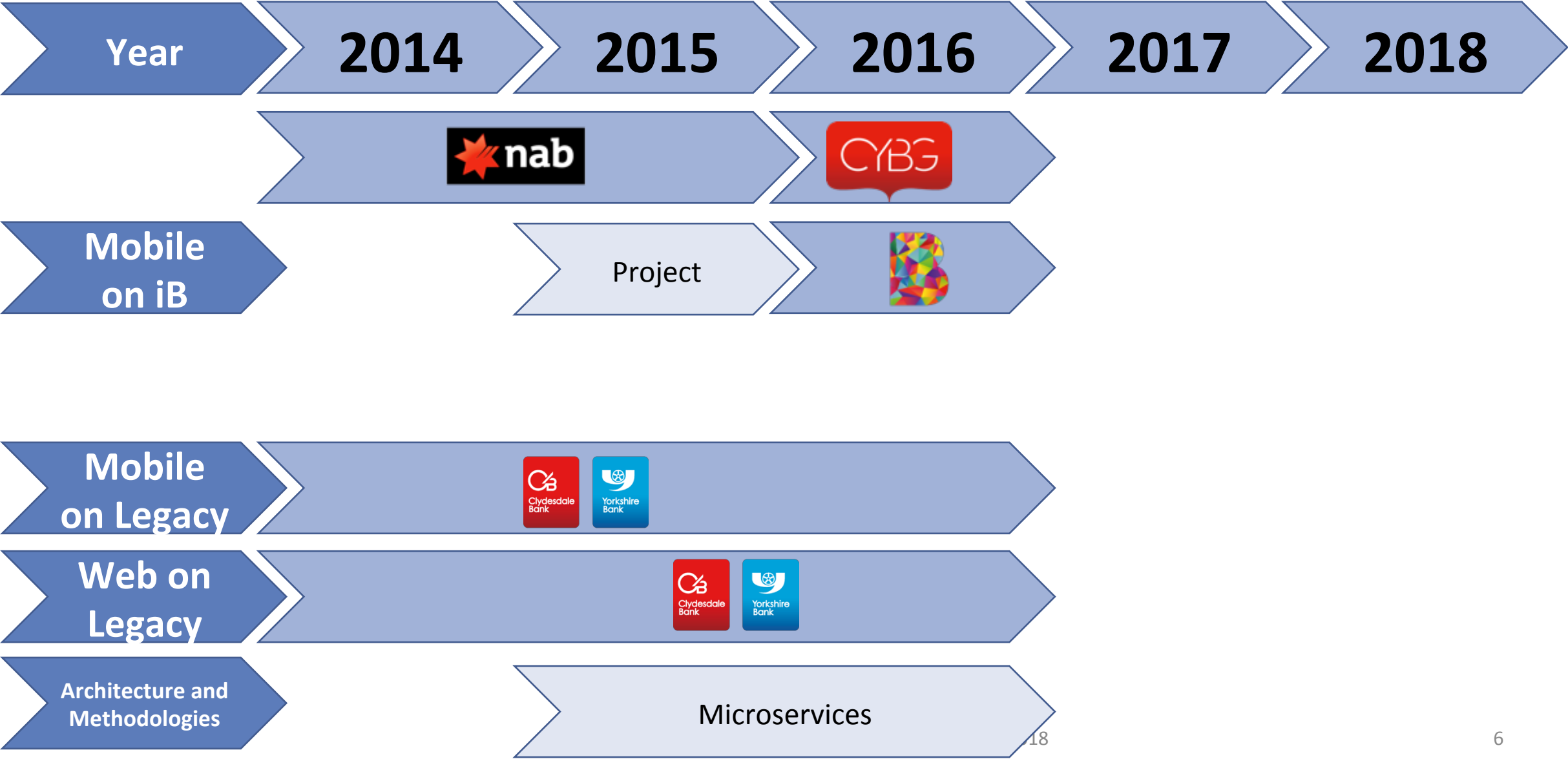- Current challenges and future plans

Who are CYBG?

- An independent UK banking group
- Listed on the LSE and ASX
- Serving customers since 1838
- Over 160 branches and a network of business and private banking centre
- 2.7m customers

Where we started

Year	2014	2015	2016	2017	2018

DYB — nab — Project

Mobile on Legacy — Clydesdale Bank, Yorkshire Bank

Web on Legacy — Clydesdale Bank, Yorkshire Bank

Architecture and Methodologies — Microservices

What did we do?

Year	2014	2015	2016	2017	2018
		nab	CYBG		
Mobile on iB		Project	B		
Mobile on Legacy		Clydesdale Bank / Yorkshire Bank			
Web on Legacy			Clydesdale Bank / Yorkshire Bank		
Architecture and Methodologies		Microservices			

What did we learn?

The Good

- Modern Architecture
- Small deployable components
- Scalability at a component level
- Component start up times
- Separation of functions
- Teams can work on different parts of the platform without getting in each others way

The Bad

- Dependency management between components was hard
- Configuration management between the different components got messy
- Versioning was the perfect mixture of art and science that was not always well understood

The Ugly

- Required Deployment Strategies (i.e still monolithic)
- Lots of effort to manage deployments
- Time taken to get 2nd and 3rd line support ready for each release
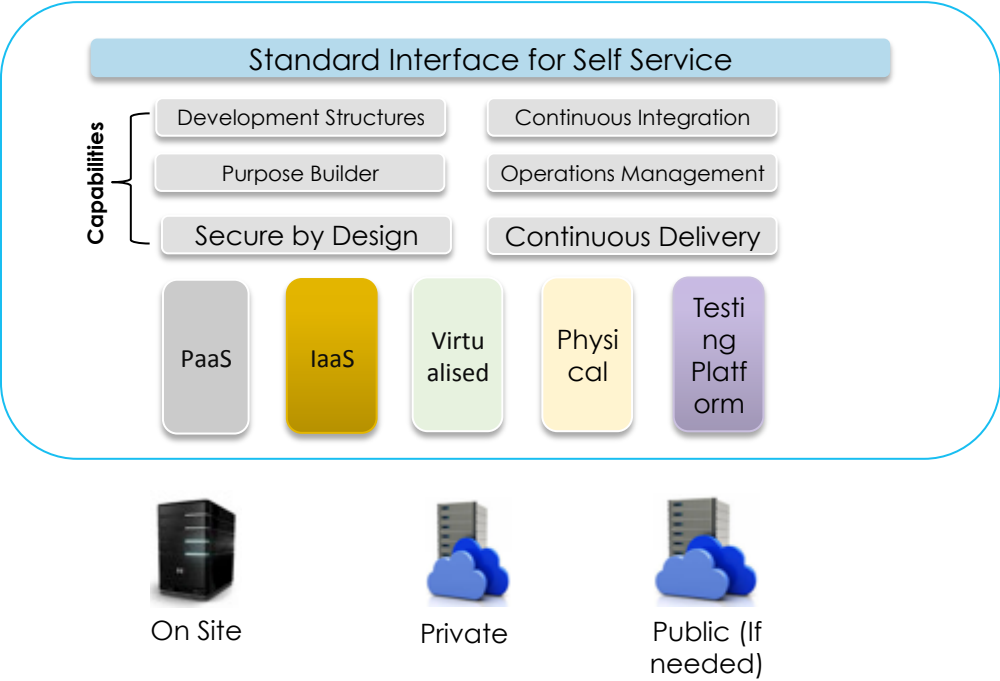- Lots of effort to mitigate the bad

What did we do?

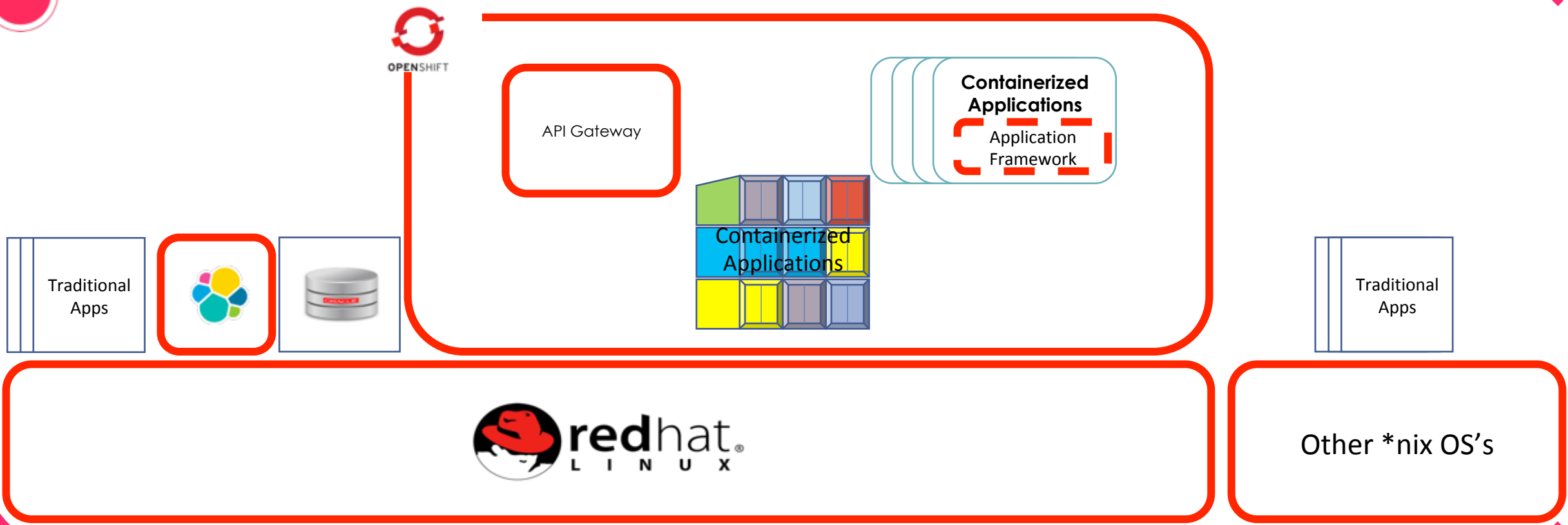| Year	2014	2015	2016	2017	2018
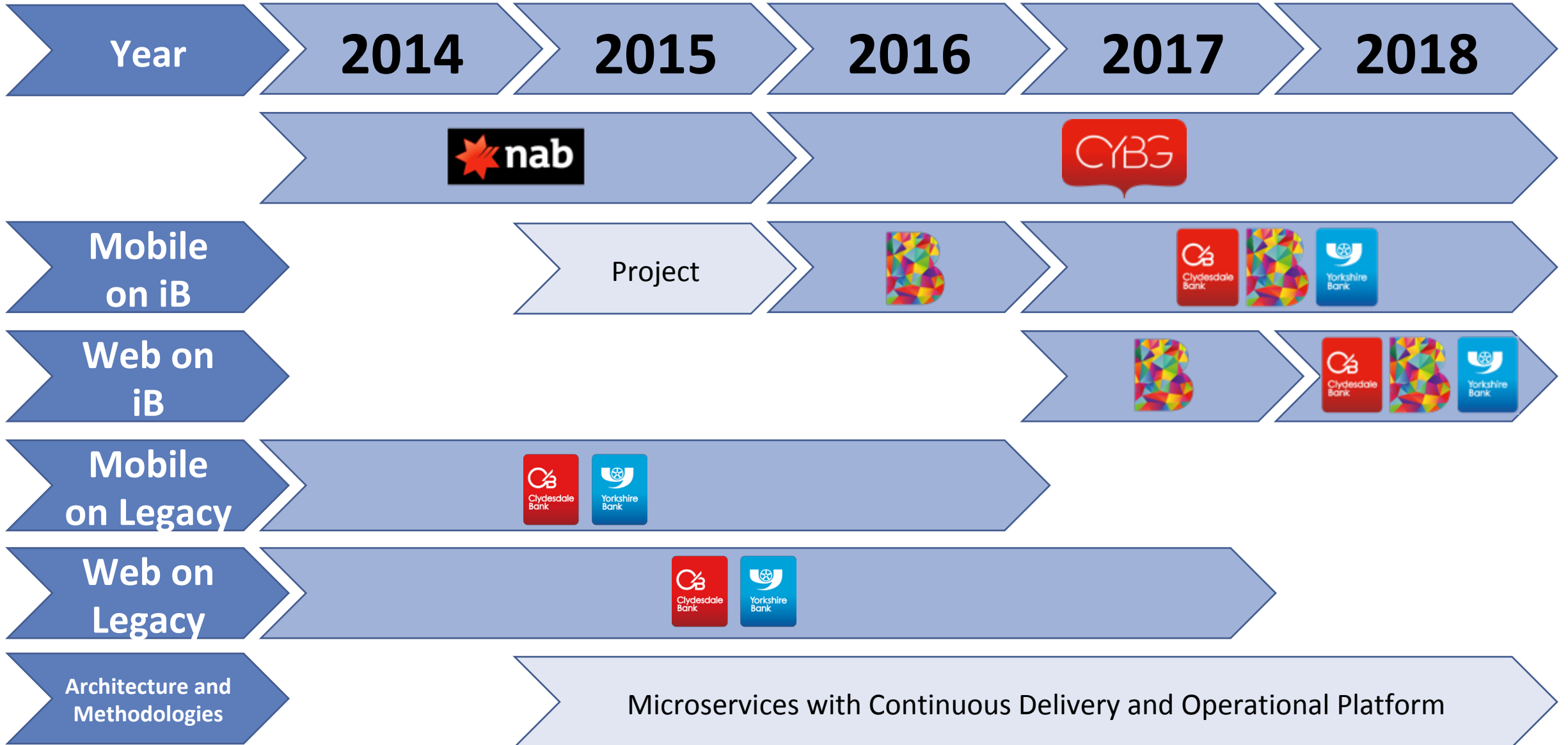
What is the Operational Platform?

CYBG

A Platform is a component which does not provide any business functionality but allows applications to be run on it

A set of capabilities designed to provide end to end management of software in a reliable, efficient and secure manner.
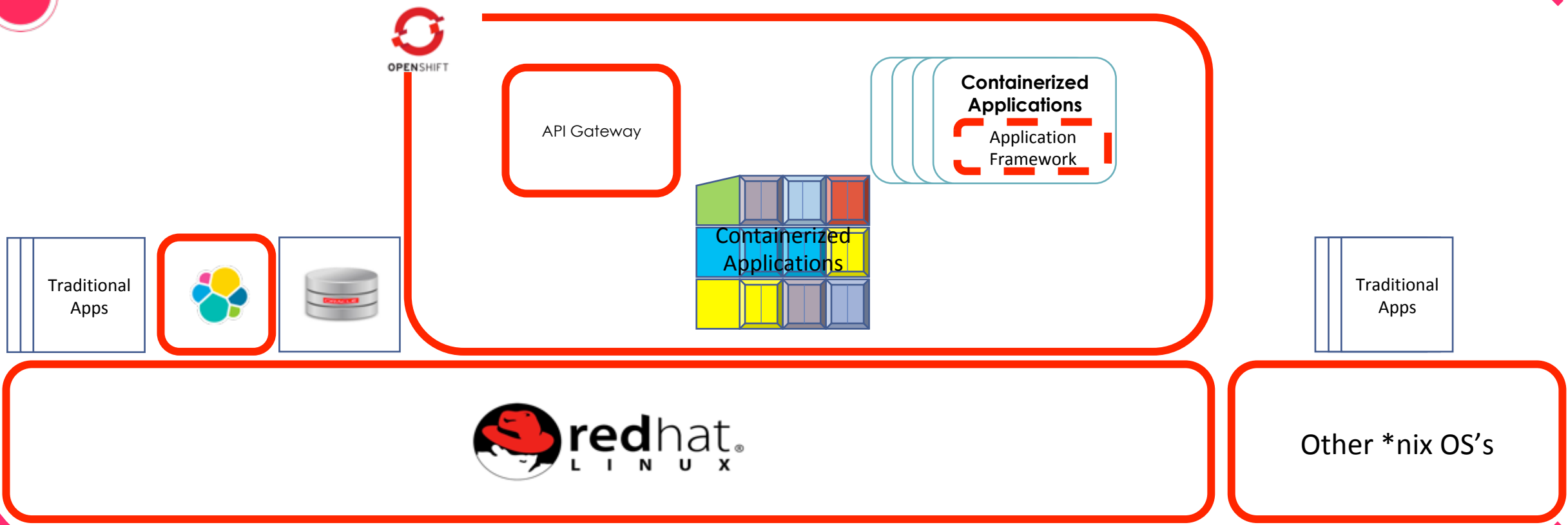

Standard Interface for Self Service

Capabilities
- Development Structures
- Purpose Builder
- Secure by Design
- Continuous Integration
- Operations Management
- Continuous Delivery

PaaS | IaaS | Virtualised | Physical | Testing Platform

On Site | Private | Public (If needed)

API Gateway

Containerized Applications

Containerized Applications
Application Framework

Traditional Apps

Traditional Apps

Containerized Applications

Other *nix OS's

redhat at the Core

Primary *nix operating system on the platform is Redhat Enterprise Linux
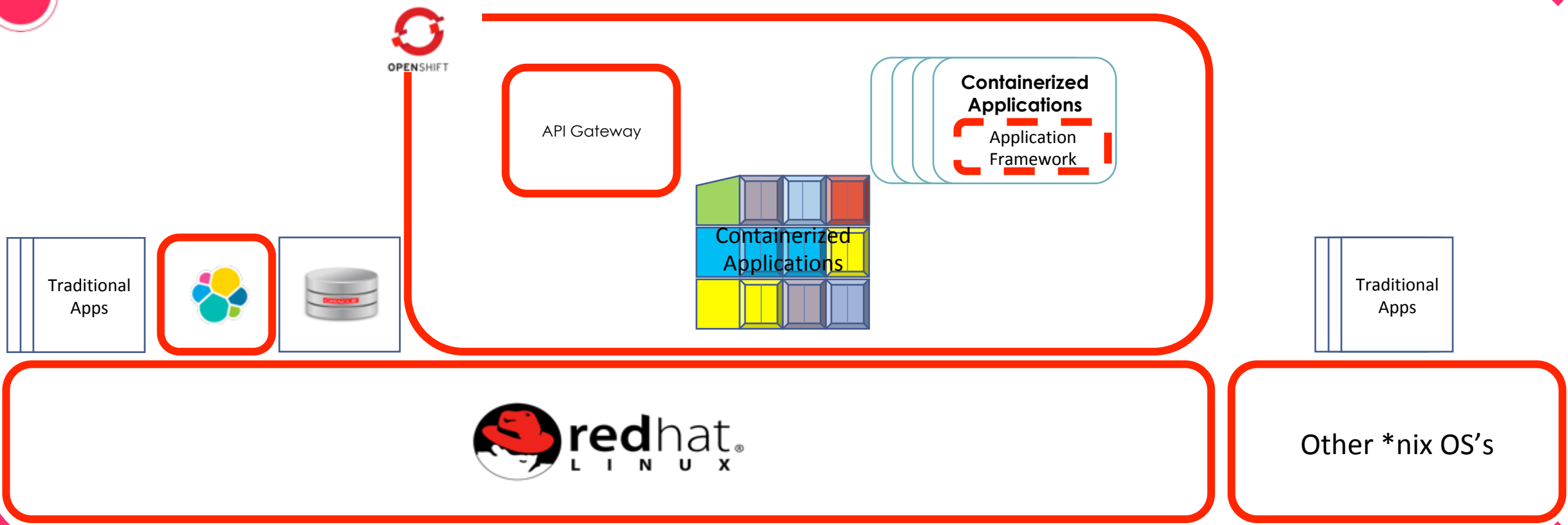
OpenShift is our Kubernetes implementation of choice.
- Implemented primarily for its operational features
  - Scaling
  - Geographical load balancing
  - Health monitoring
  - Templates

Satellite and Cloud forms used for provisioning and management of our estate.

Nexus

sonarqube

APPDYNAMICS

kibana

iCINGA

CHIEFTAIN

ELEMENTS

API Gateway

Containerized Applications

Application Framework

Containerized Applications

Traditional Apps

Traditional Apps

redhat LINUX

Other *nix OS's

What is Chieftain?

System of record for storing component configuration which can be provided in a standard format.

Use of JSON to return information in a structured format.

Auto population of dependency information without having to know all the details about dependencies.

Common Questions

What about all the other "Configuration Managers", Chef, Ansible, Puppet, Salt Stack etc?

What about all the CMDB providers?
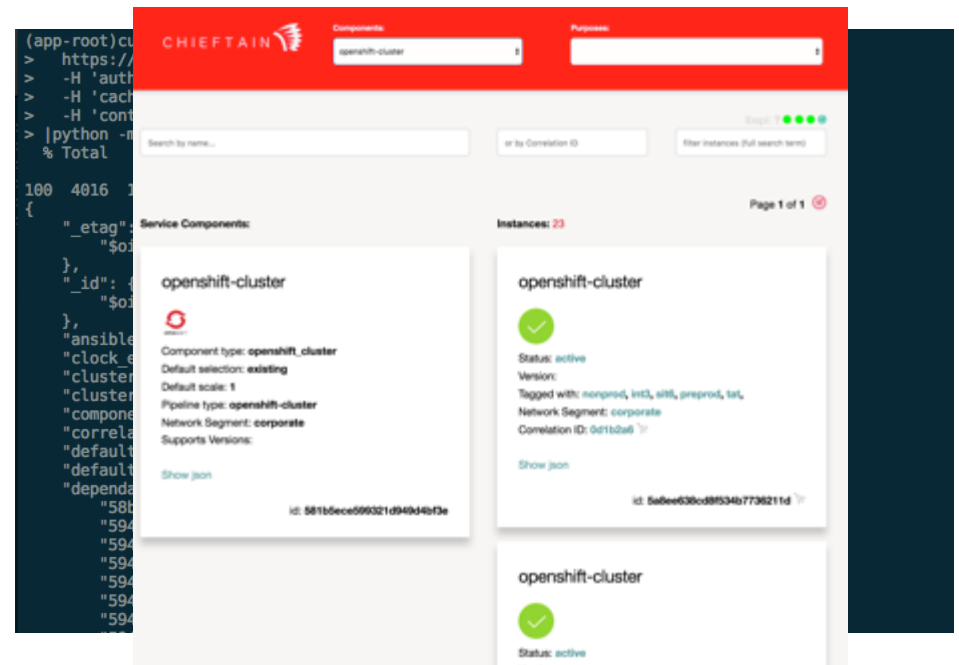
Composer

What is composer?

Environment builder, deploy orchestrator.
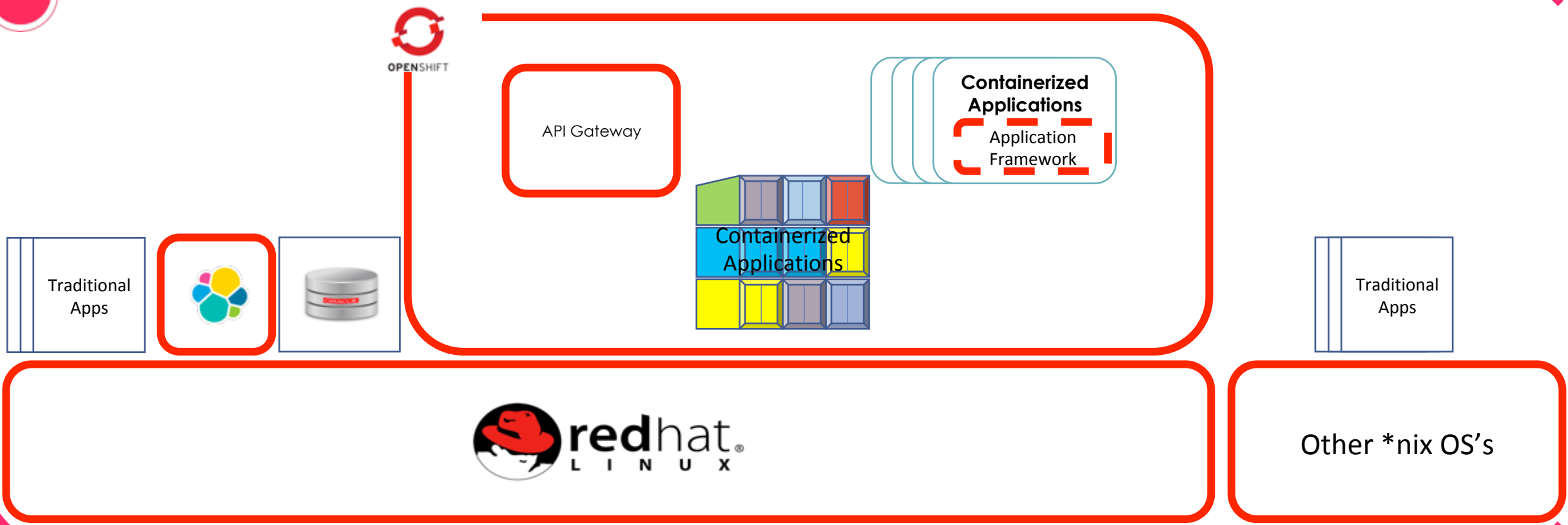Component Dependency Resolver

What are elements?

Library of scripts used by Composer to build and deploy
to any environment or platform.

ELEMENTS

OPENSHIFT

Nexus
sonarqube
go go go

APPDYNAMICS
kibana
ICINGA

OPENSHIFT

API Gateway

Containerized
Applications
Application
Framework

Containerized
Applications

Traditional
Apps

Traditional
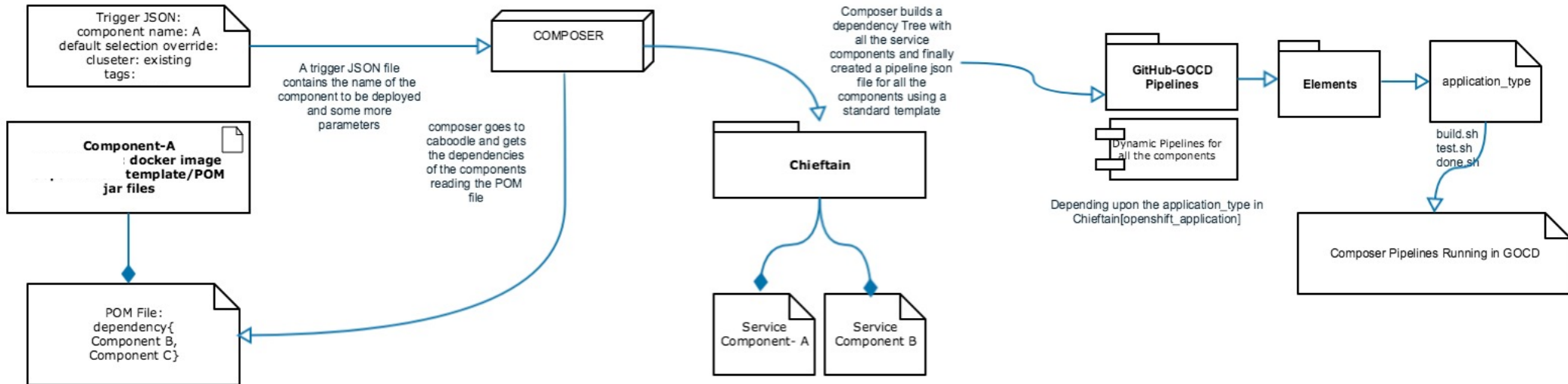Apps

redhat
LINUX

Other *nix OS's

Continuous Integration and Delivery with Composer

	CI Builds – Build PR's and Release Candidates Every build gets its own full functioning environment to run Developer written Integration Tests
Nexus	Storage of artifacts as well as opensource vulnerability scanning
sonarqube	Code Quality Checks
	CD - Route to live pipelines on a per component basis

Continuous Delivery Pipelines

API Gateway

Containerized Applications

Containerized Applications

Application Framework

Traditional Apps

Containerized Applications

Traditional Apps

Other *nix OS's
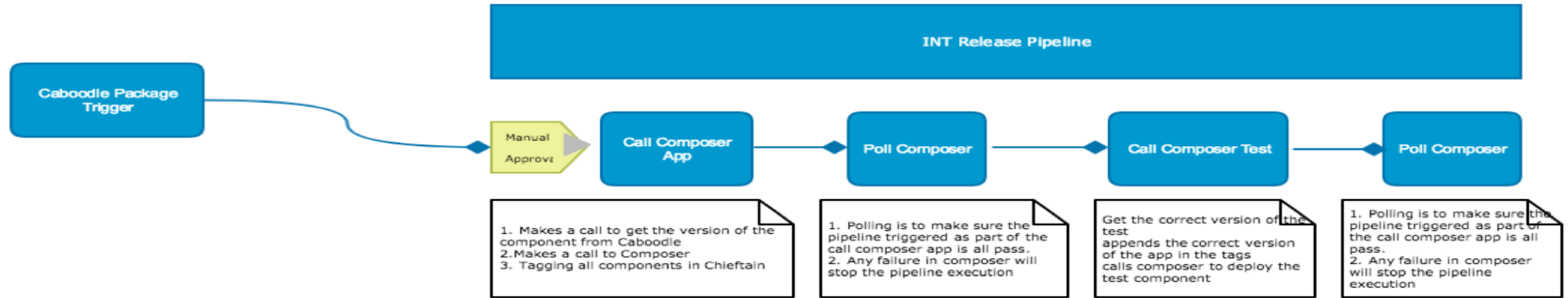
ELEMENTS
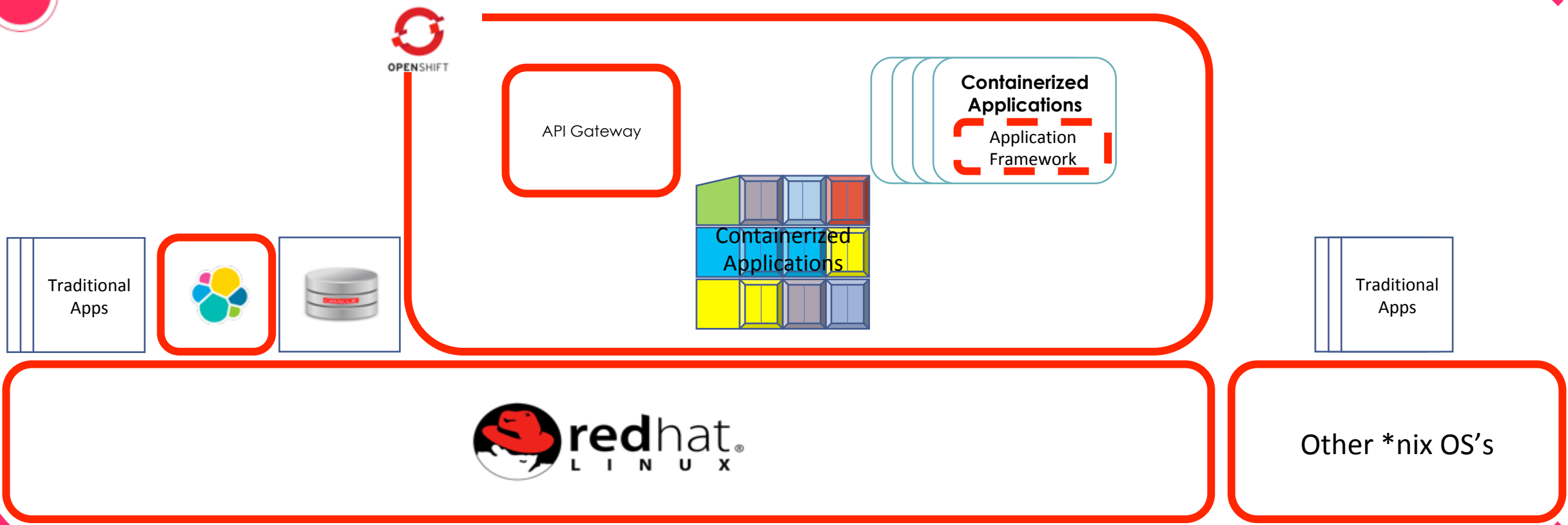
API Gateway and Application Frameworks

API Gateway

- Built on Open Source technology from Spring

- Took inspiration from Fabric8 for integrating with OpenShift to get service location

- Dynamically registers Microservices as they start taking event from Kubernetes then interrogating microservice for details about itself

Application Frameworks

- Based on Open Source technology
  - Java
  - Python and JS

- Integrates tightly with Kubernetes to provide operational imperatives
  - Restarts, liveliness, health, metrics

- Integrates with API Gateway

- Monitoring bundled in as part of build

- Component dependency management out of the box

- Standard repo structure

What does all this mean for delivery?

Delivery of a Prototype in 2017

52 Builds

89 UAT Releases

36 Production Releases

3 Months

Delivery of a Prototype in 2018
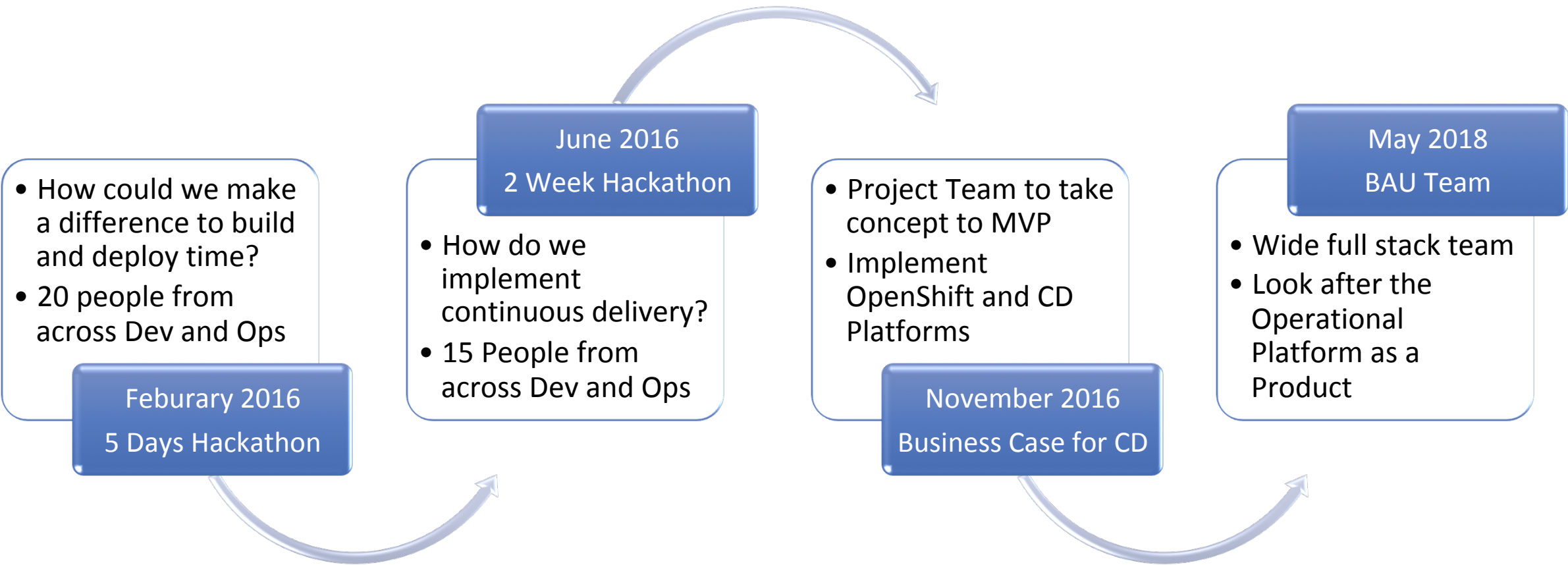
94 Builds - (99 Integration Deploys)

74 UAT Releases

0 Production Releases

1 Month

Latency during deployments is now measured in minutes ☺

Evolution of the Team

CYBG

- How could we make a difference to build and deploy time?
- 20 people from across Dev and Ops

Feburary 2016
5 Days Hackathon

June 2016
2 Week Hackathon

- How do we implement continuous delivery?
- 15 People from across Dev and Ops

- Project Team to take concept to MVP
- Implement OpenShift and CD Platforms

November 2016
Business Case for CD

May 2018
BAU Team

- Wide full stack team
- Look after the Operational Platform as a Product

How Redhat helped us

- Professional Services
    - Architecture – Helped us get the design right, then validated once it was implemented
    - Engineering – On site engineer at various different times, helped with automation but also helped out across the project as needed – really bought in to our self managed team
    - On going support through the usual support channels

- Continually providing us with new ideas about different technologies
    - Through blogs and publications
    - Attending Redhat conferences
    - On going and regular relationship discussions

- Encouraged us to take part in opensource
    - Showing the art of the possible

- Inspiring us with their use of Open Organization

What Next?

CYBG - Engineering The Bank - Redhat Forum London 2018

Thank you for your time.
Questions?