Redhat Forum – London 2018

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

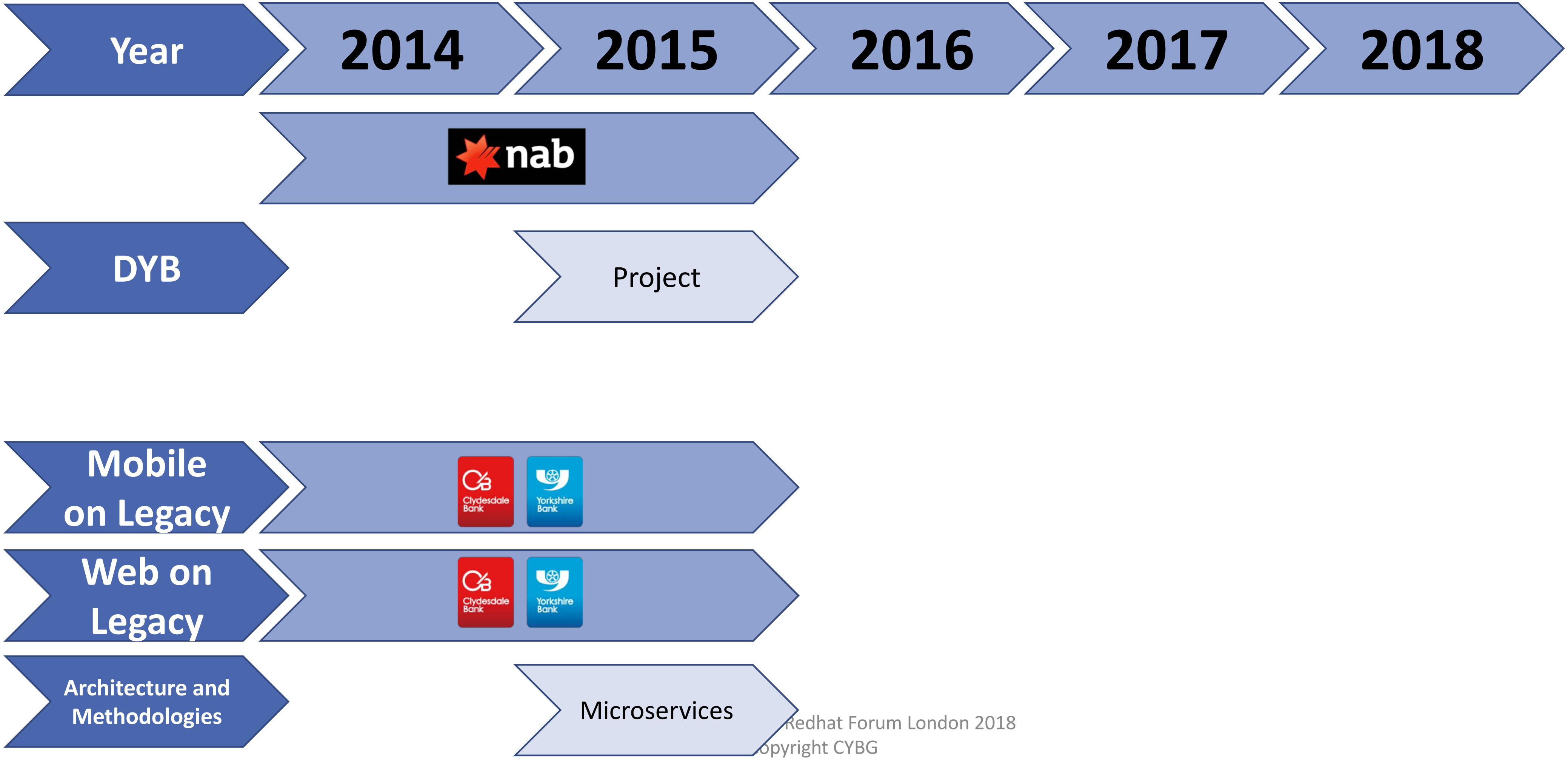
What will we cover?

- 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?

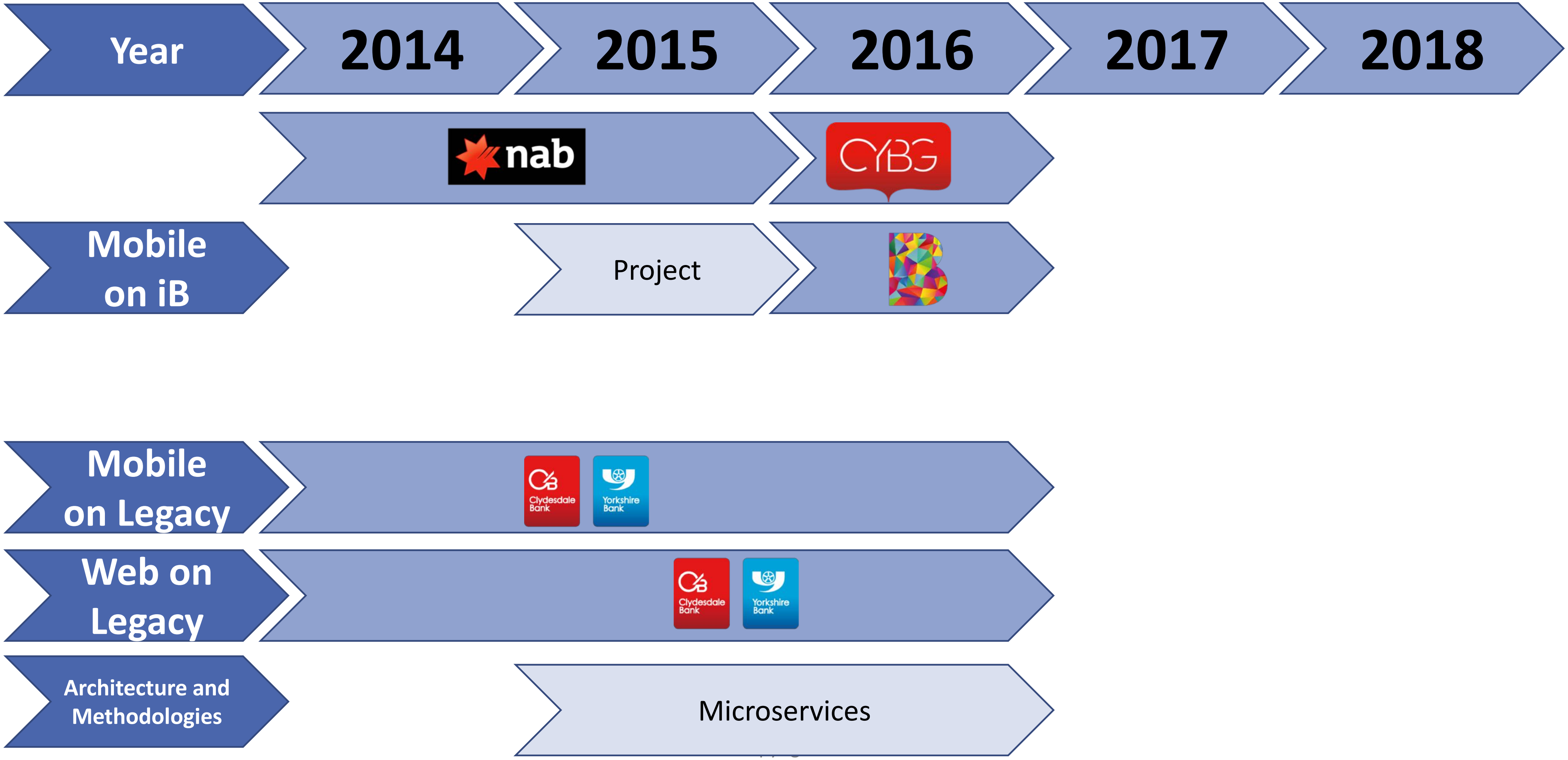
- An independent UK banking group
- Listed on the LSE and ASX
- Serving customers since 1838
- Over 160 branches and a network of business and private banking centre
- 2.7m customers

Where we started



Year	2014	2015	2016	2017	2018
	nab				
DYB		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

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

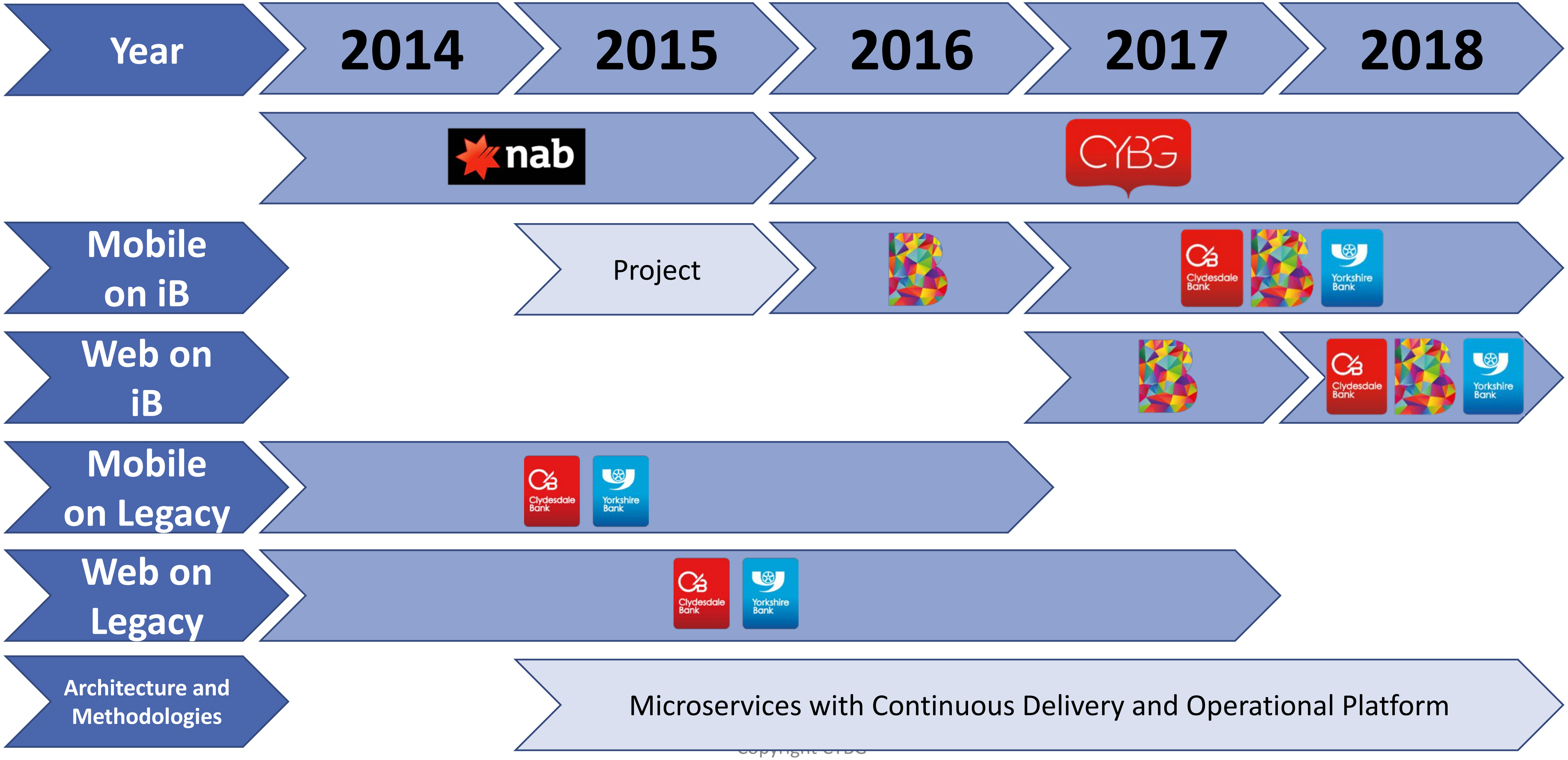
😢 The Bad

- 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

Mobile on iB

Project

Web on iB

Mobile on Legacy

Web on Legacy

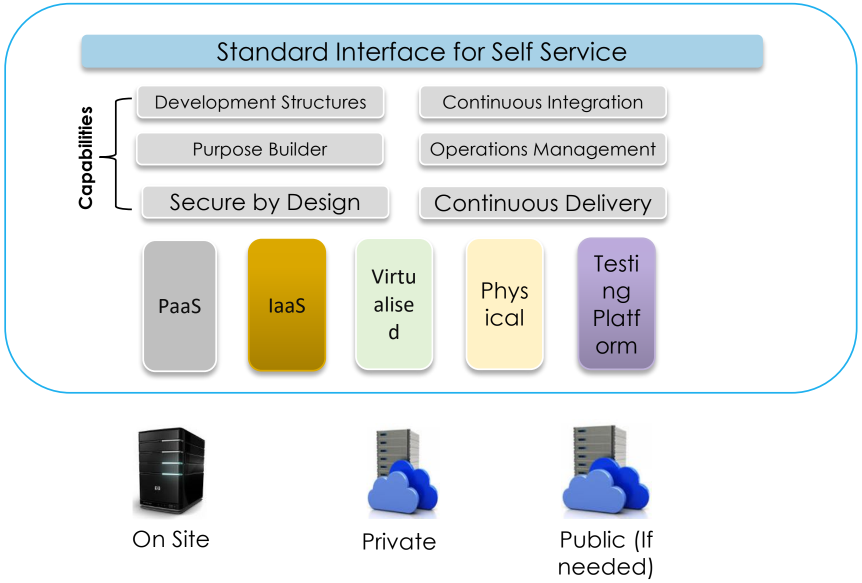
Architecture and Methodologies

Microservices with Continuous Delivery and Operational Platform

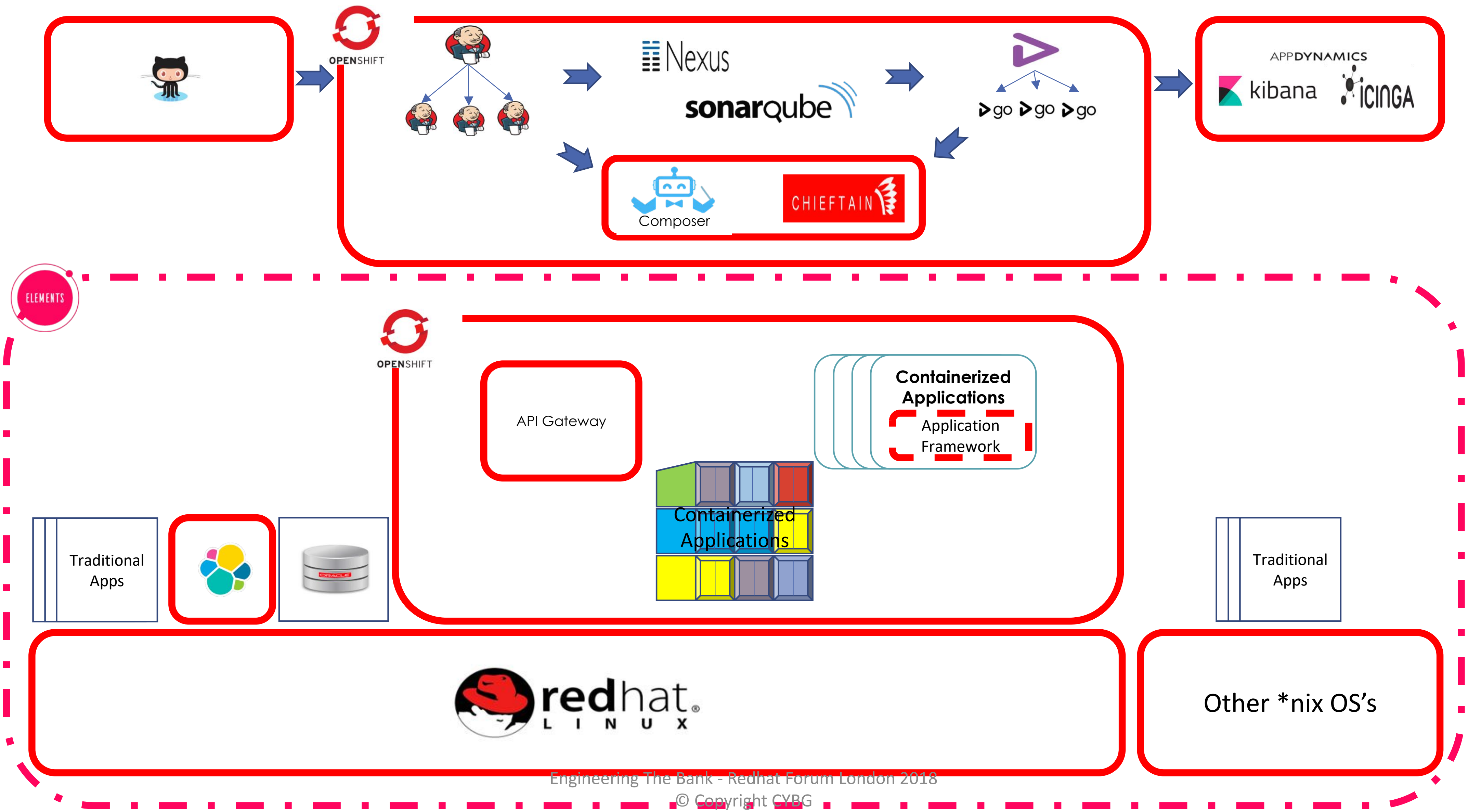
What is the Operational Platform?

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.



Engineering The Bank - Redhat Forum London 2018



API Gateway

Containerized Applications

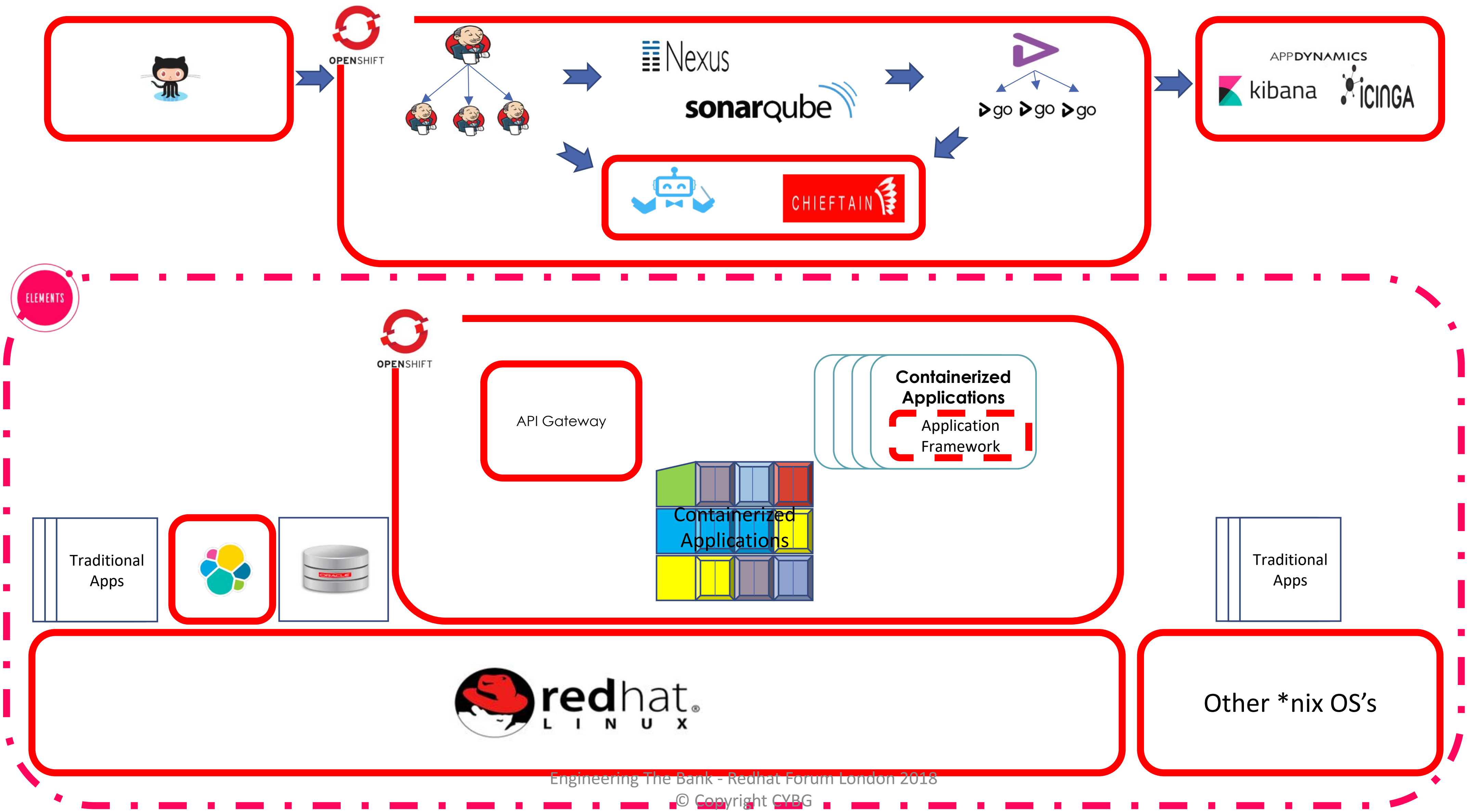
Application Framework

Containerized Applications

Traditional Apps

Traditional Apps

Other *nix OS's



API Gateway

Containerized Applications

Application Framework

Containerized Applications

Traditional Apps

Traditional Apps

Other *nix OS’s

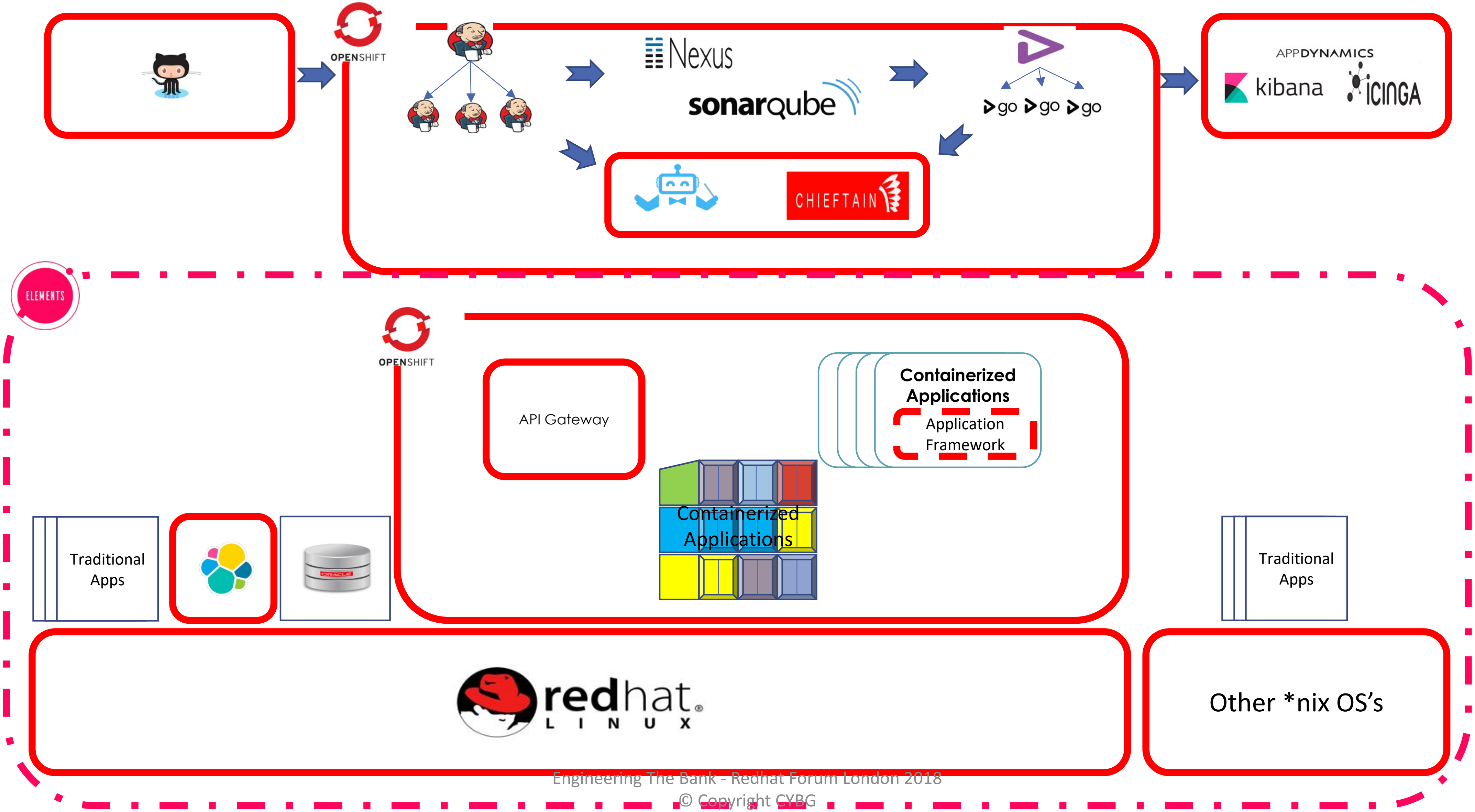
redhat at the Core

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 Cloudforms used for provisioning and management of our estate.



API Gateway

Containerized Applications

Application Framework

Containerized Applications

Traditional Apps

Traditional Apps

Other *nix OS's

CHIEFTAIN

What is Chieftain?

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

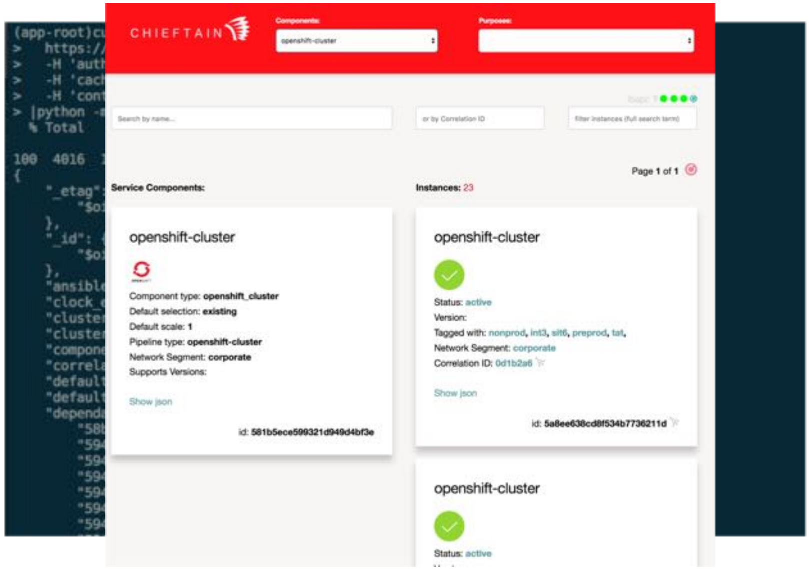
Use of JSON to return information in a structured format.

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?



```
(app-root)cu
>    https://
>    -H 'auth
>    -H 'cach
>    -H 'cont
> |python -m
  % Total

100  4016  1
{
    "_etag":
        "$oi
    },
    "_id": {
        "$oi
    },
    "ansible
    "clock_c
    "cluster
    "cluster
    "compone
    "correla
    "default
    "default
    "depende
        "58b
        "594
        "594
        "594
        "594
        "594
        "594
```

Page **1** of **1**

Service Components:

openshift-cluster

Component type: **openshift_cluster**
Default selection: **existing**
Default scale: **1**
Pipeline type: **openshift-cluster**
Network Segment: **corporate**
Supports Versions:

Show json

id: **581b5ece599321d949d4bf3e**

Instances: 23

openshift-cluster

Status: active
Version:
Tagged with: nonprod, int3, sit6, preprod, tat,
Network Segment: corporate
Correlation ID: 0d1b2a6

Show json

id: **5a8ee638cd8f534b7736211d**

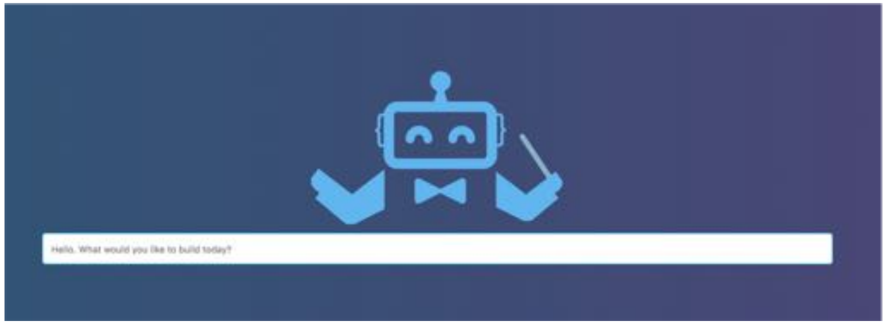
openshift-cluster

Status: active

Composer & Elements

What is composer?

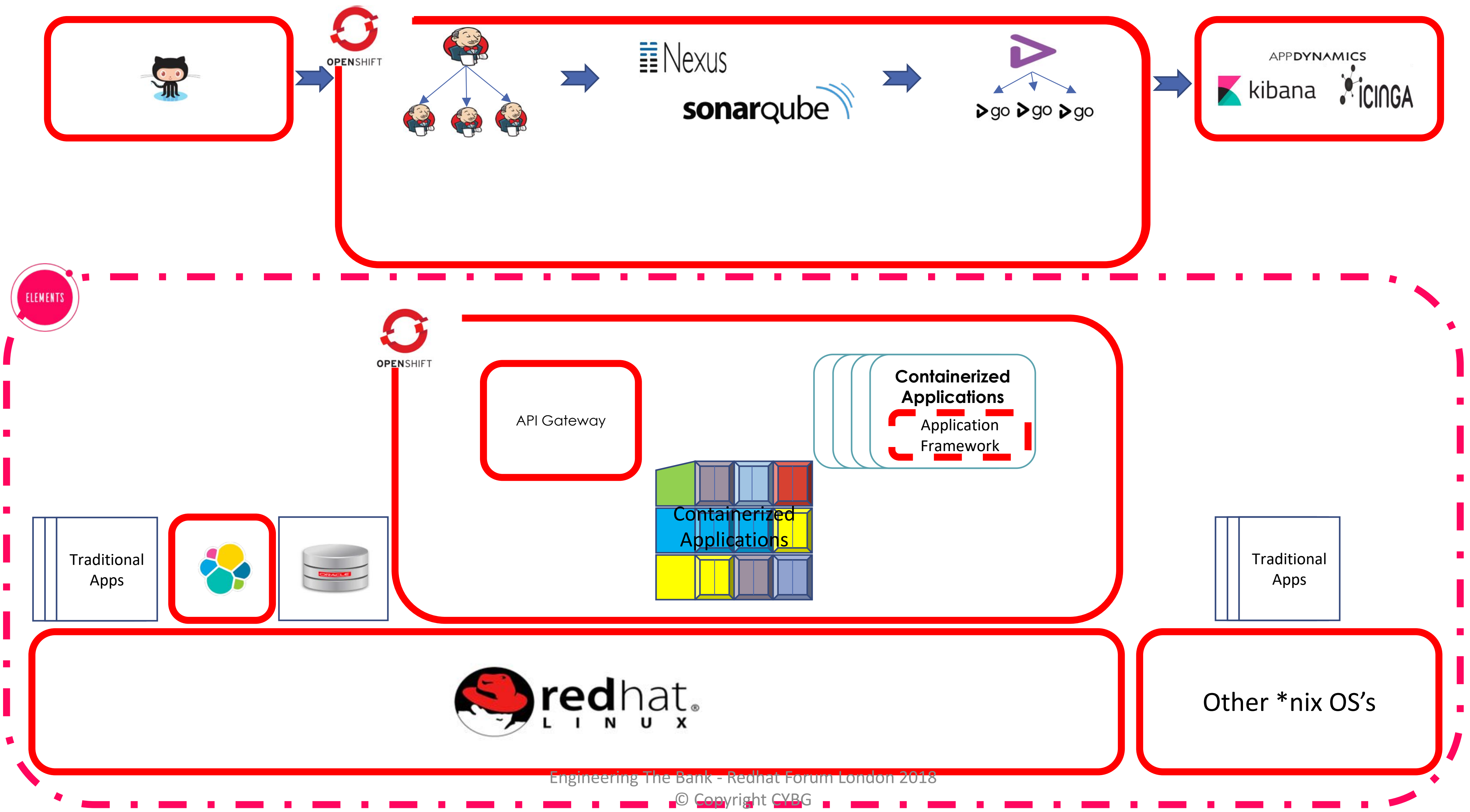
Environment builder, deploy orchestrator.
Component Dependency Resolver



Hello. What would you like to build today?

What are elements?

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



API Gateway

Containerized Applications

Application Framework

Containerized Applications

Traditional Apps

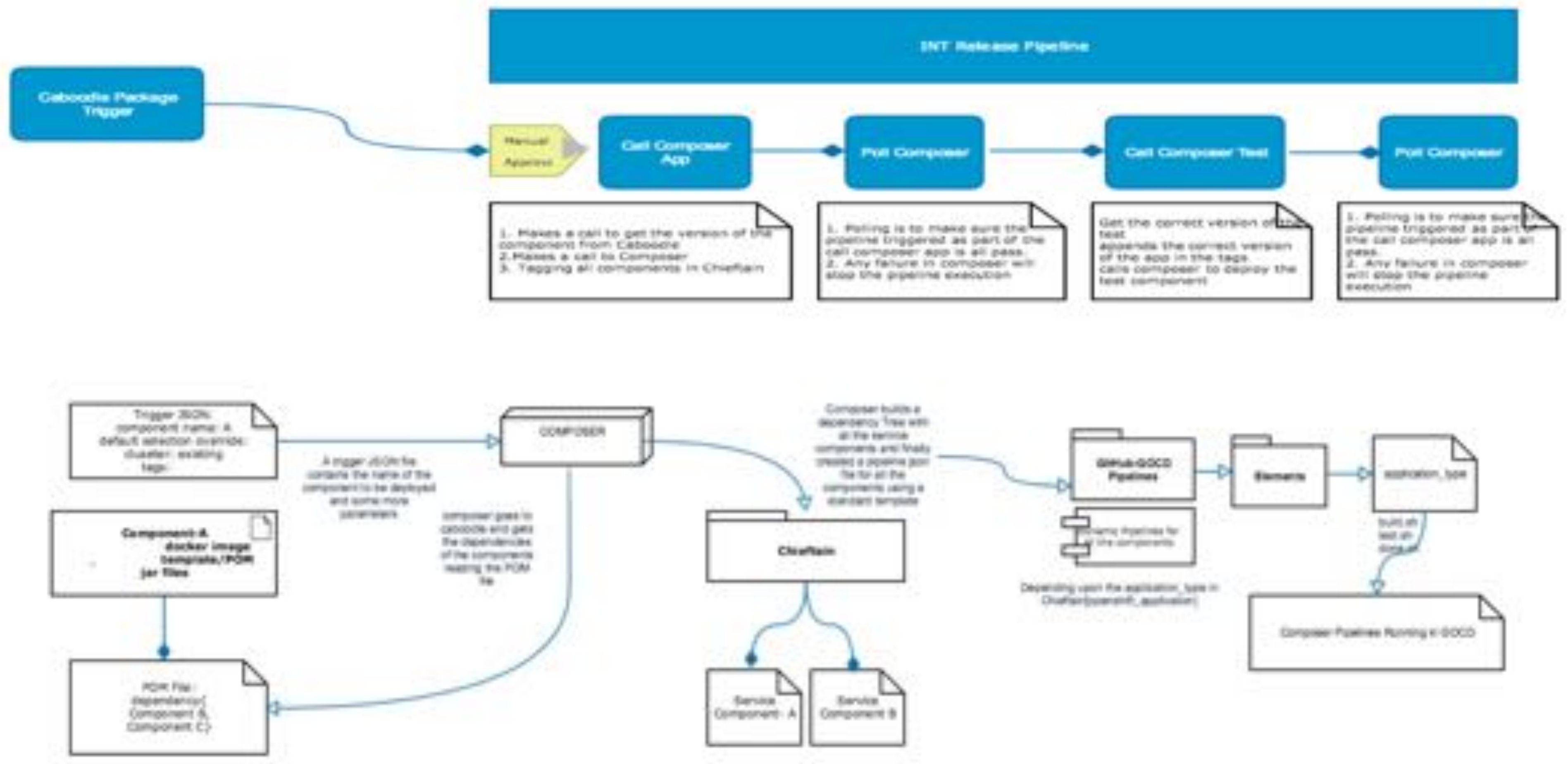
Traditional Apps

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



INT Release Pipeline

Caboodle Package Trigger

Manual Approval

Call Composer App

Poll Composer

Call Composer Test

Poll Composer

1. Makes a call to get the version of the component from Caboodle
2. Makes a call to Composer
3. Tagging all components in Chieftain

1. Polling is to make sure the pipeline triggered as part of the call composer app is all pass.
2. Any failure in composer will stop the pipeline execution

Get the correct version of the test
appends the correct version of the app in the tags
calls composer to deploy the test component

1. Polling is to make sure the pipeline triggered as part of the call composer app is all pass.
2. Any failure in composer will stop the pipeline execution

Trigger JSON: component name: A default selection override: cluster: existing tags:

A trigger JSON file contains the name of the component to be deployed and some more parameters

COMPOSER

composer goes to caboodle and gets the dependencies of the components reading the POM file

Component-A docker image template,/POM jar files

POM File: dependency{ Component B, Component C}

Composer builds a dependency Tree with all the service components and finally created a pipeline json file for all the components using a standard template

Chieftain

Service Component - A

Service Component B

GitHub-GOCD Pipelines

Dynamic Pipelines for all the components

Depending upon the application_type in Chieftain(openshift_application)

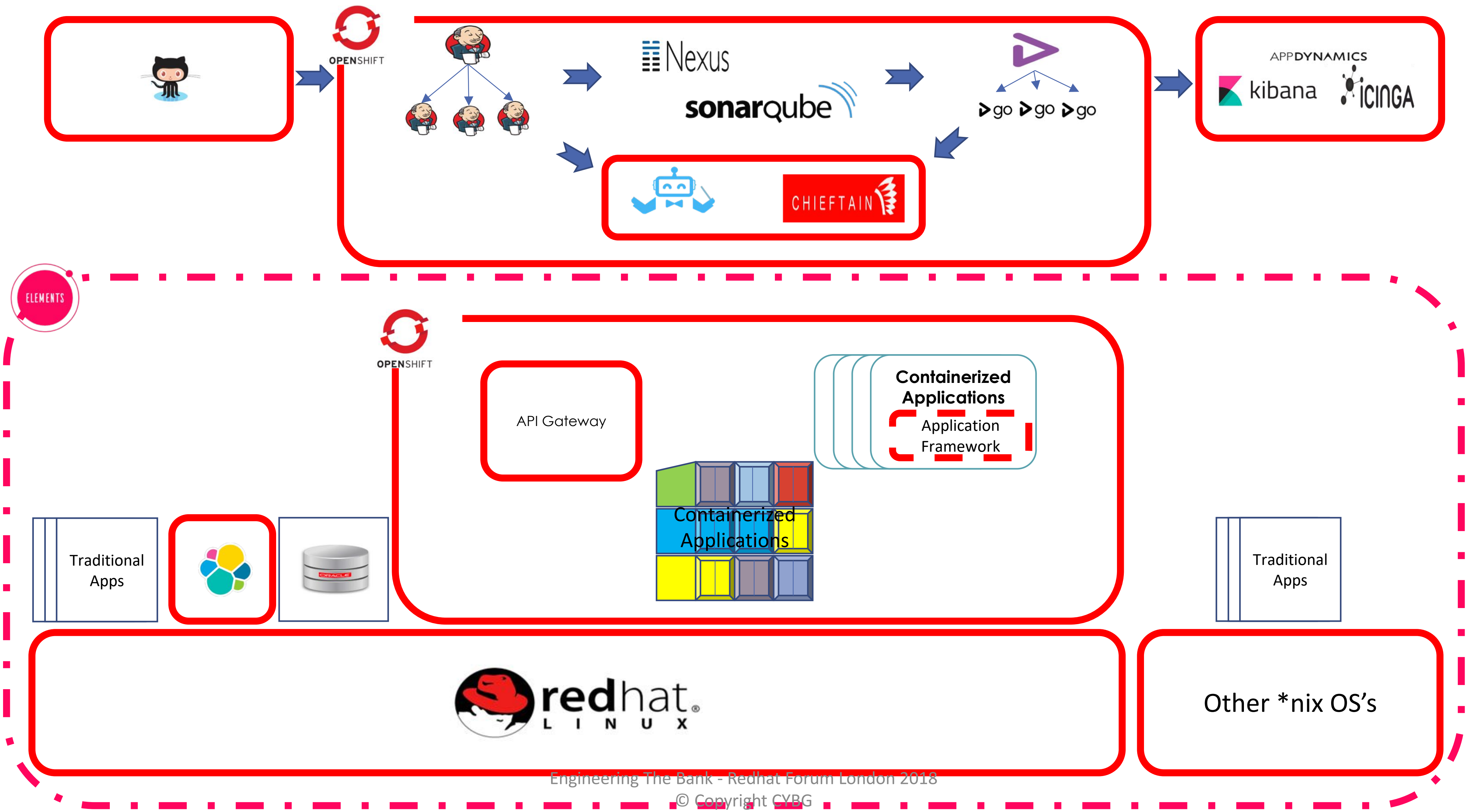
Elements

application_type

build all test all done all

Composer Pipelines Running in GOCD

Engineering The Bank - Redhat Forum London 2018


OPENSHIFT
Nexus
sonarqube
go go go
CHIEFTAIN
APPDYNAMICS
kibana
icinga
ELEMENTS
OPENSHIFT
API Gateway
Containerized Applications
Application Framework
Containerized Applications
Traditional Apps
Traditional Apps
ORACLE
redhat LINUX
Other *nix OS's

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)

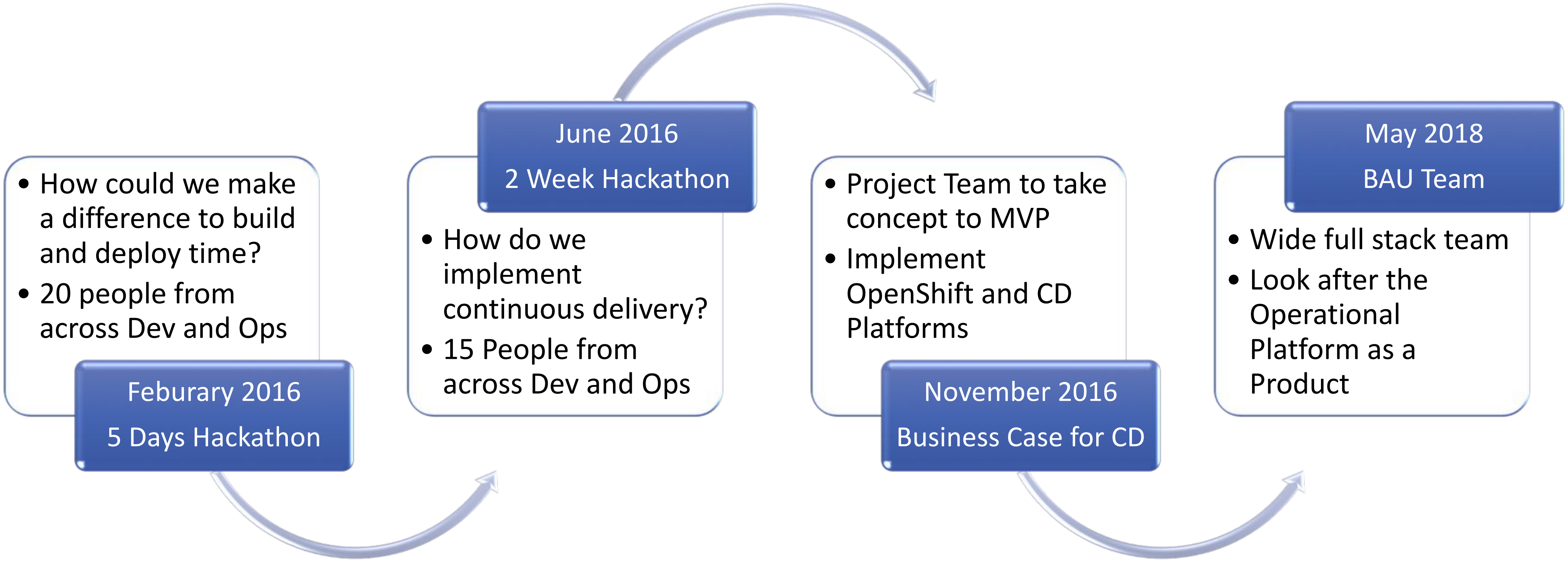
74 UAT Releases

0 Production Releases

1 Month

Latency during deployments is now measured in minutes ☺

Evolution of the Team



- 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?

Thank you for your time.
Questions?

Engineering The Bank - Redhat Forum London 2018