

# ARCHITECTURE EVOLUTION

FROM AMORPHOUS MONOLITH  
APPLICATION TO A MODULAR, CLOUD  
READY PLATFORM

Coralia Popa – April 2020



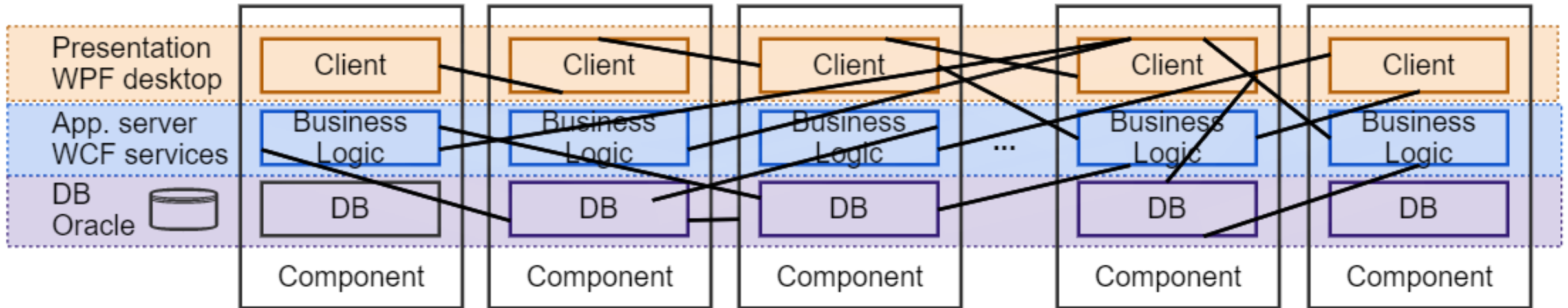
# Agenda

- The monolith
- Layered architecture
- Applications segregation
- Service based modular architecture
- Switch to dual deployment: on premise and cloud
- Final thoughts

# The monolith in numbers

- 5 years ago
- 15 years old application at that time
- 2 years release cycle
- 100+ non collocated engineers
- 3 sources of complexity: integration with analytical instruments, regulatory compliance, specific data processing algorithms
- 2 types of deployment: workstation, network
- 3 tier architecture: database, business logic and presentation
- 3 paths leading to monolith : code management, deployment, runtime
- 50+ vertical slice components

# The monolith as a diagram



# The monolith as in a list of problems



EXTREMELY **LOW SPEED**  
OF FEATURE DELIVERY



BIG **RISK OF**  
**REGRESSION** DURING  
MAINTENANCE



REDUCED OPPORTUNITY  
TO CHANGE  
**TECHNOLOGIES**



PERFORMANCE NEVER  
ALLOWED **SCALING** TO  
THE NEEDED  
PARAMETERS



**INDUSTRY TRENDS**  
INCOMPATIBILITY



The turning  
point



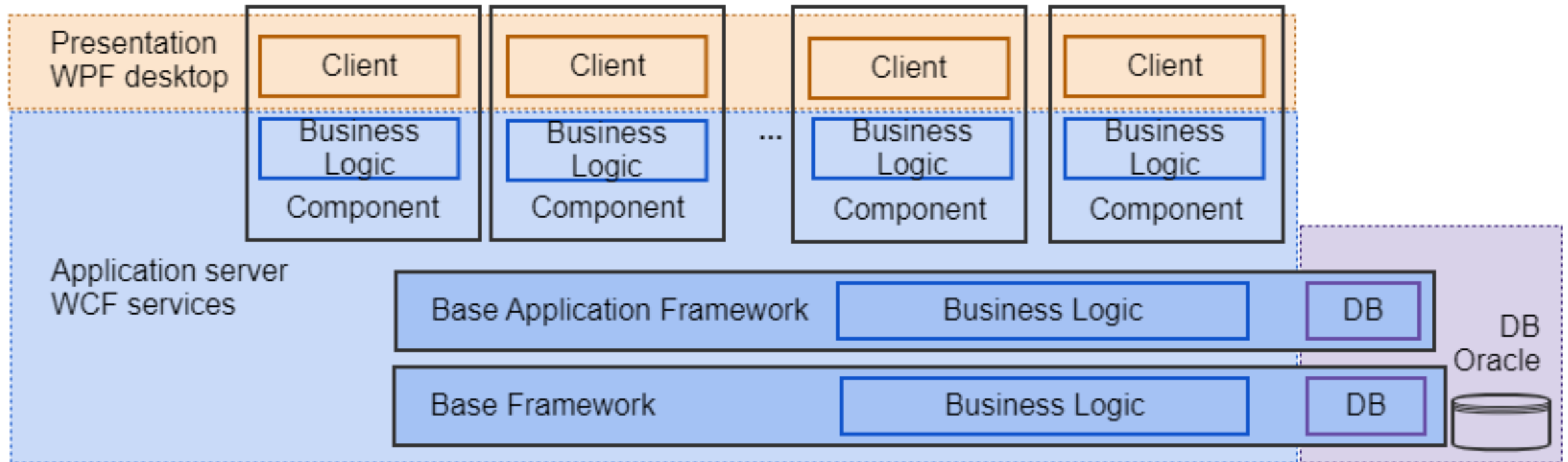
# Decision point

## Incremental evolution through refactoring or Big Bang rewrite?

- Improved componentization through refactoring
  - Break the “code” monolith by layered architecture for business logic
- Gradual replacement with rewrite



# Layered architecture



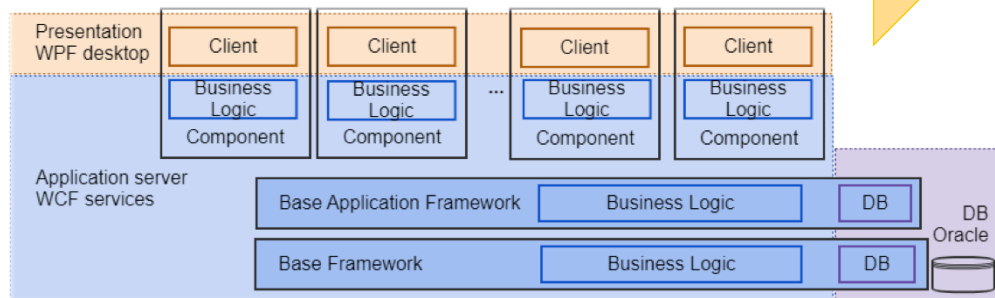


# Layered architecture - Believes and hopes

- Core software principles power

- Reuse-release Equivalence
- Common-Reuse
- Stable-Dependencies
- Acyclic Dependencies

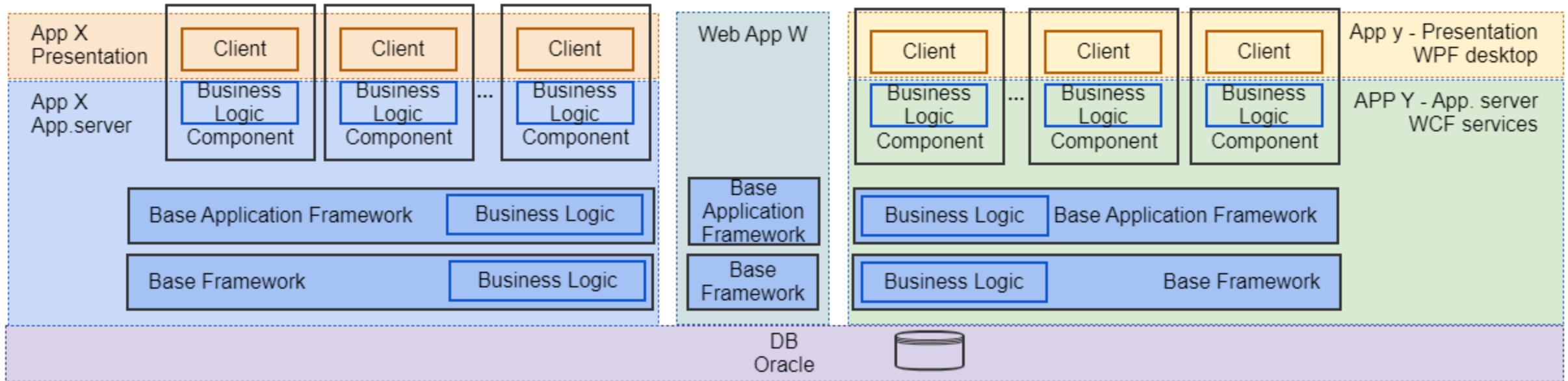
- bottom layers can be **stabilized**
- better understanding of **change impact**
- simpler control of **dependencies**
- better **testability** of the business logic
- enable the presentation to evolve at **different change pace**



# First signs of recovery

- Could release twice with the same version of the bottom layers
- The rate of changes on the bottom layers on descendent trend
- Development of new application could be started on top of the new layers
- But...
  - Demoralized engineer teams given the massive effort with not much satisfaction
  - Hard to demonstrate to the management the outcome of investment

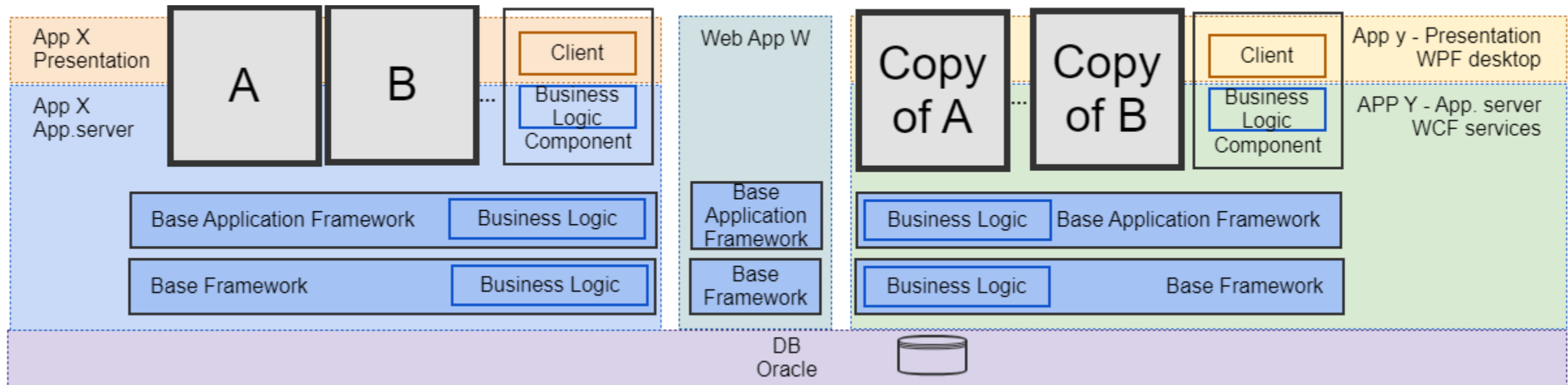
# Next step in evolution - Break down the “runtime monolith”



# Decision point

## DRY versus LOOSE COUPLING dilemma

- Dramatic and unpopular decision to copy a large amount of code just to step in the right direction

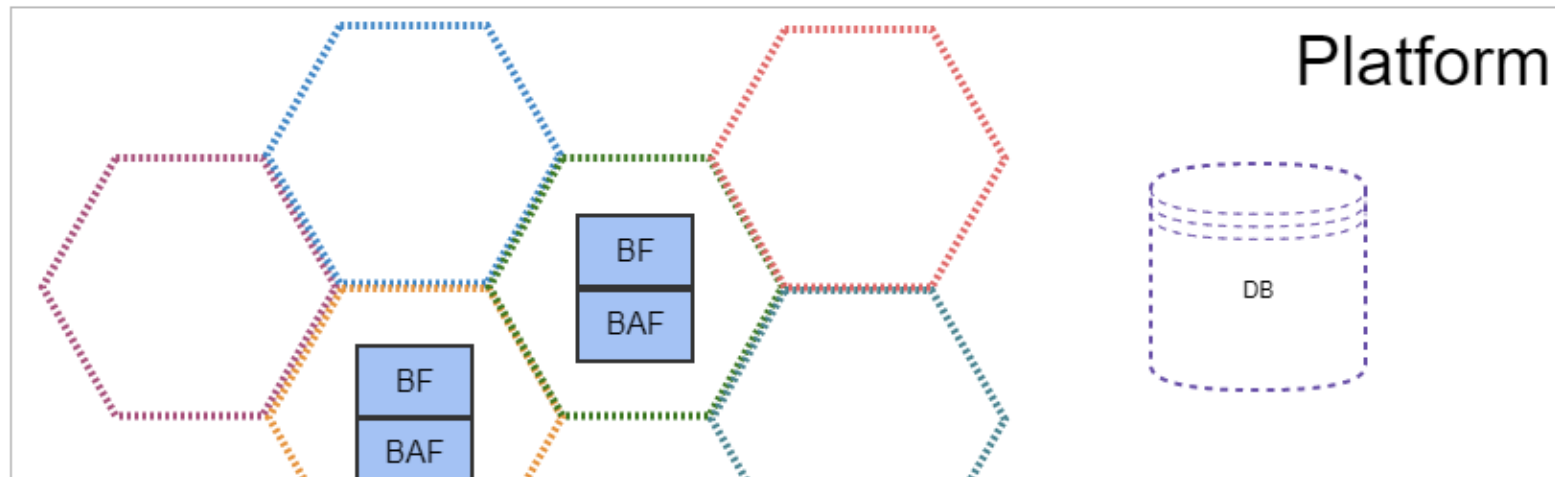
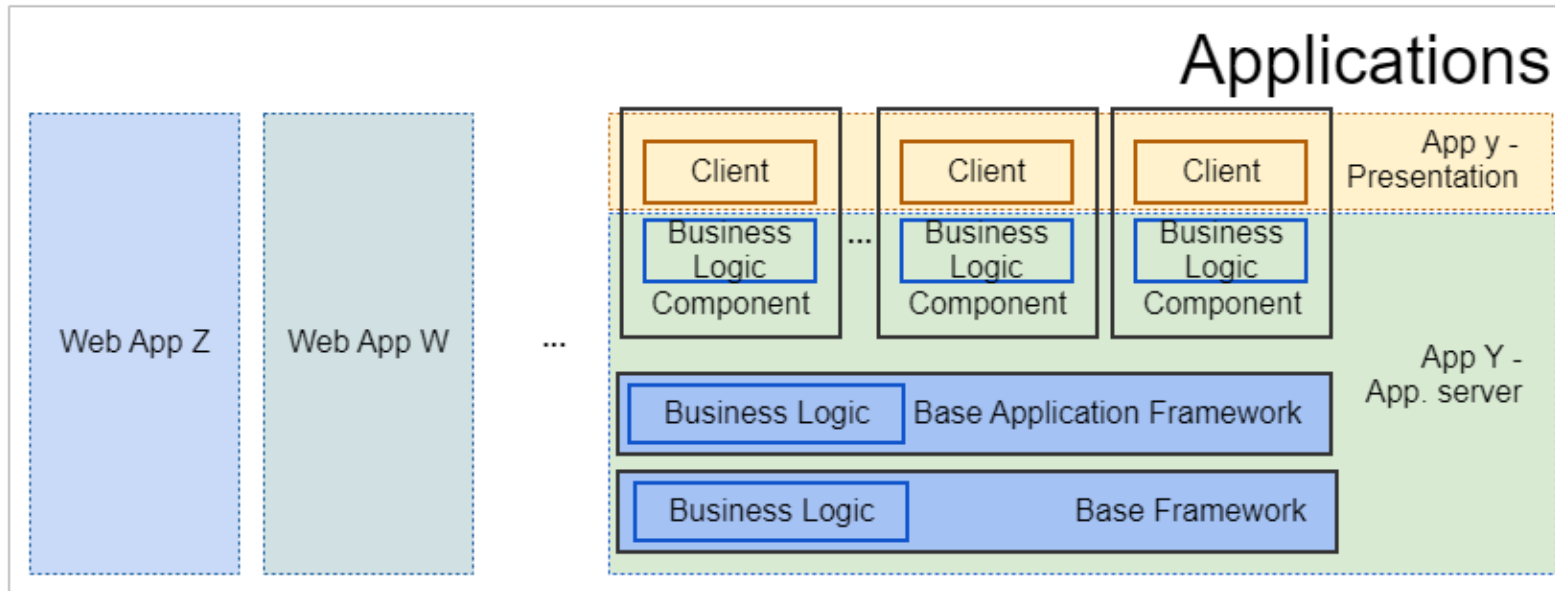


# Benefits

- Almost impossible to add coupling between application apps
- Distribution of the load in different processes helped in some extent with performance
- Release cadence reduced to a quarter already, the original objective was reached

But

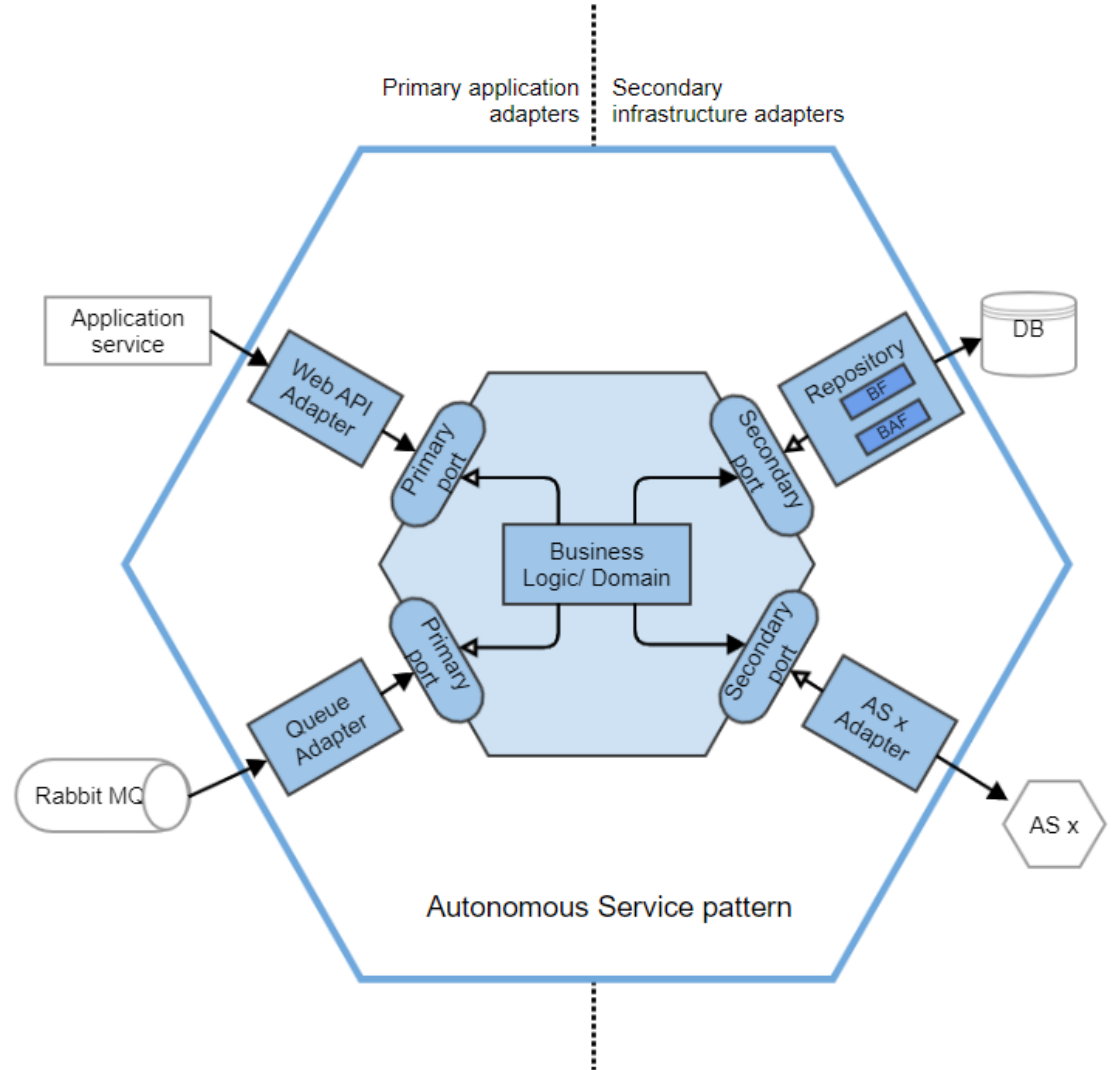
- Started to have issues keeping the instances of the layers in sync between apps



Next step in evolution -  
Break the deployment monolith

# Autonomous service pattern

- covers a domain functionality
- loosely-coupled components following hexagonal architecture
- easy adoption as is popular architecture
- wide set of testing options
- no state
- clear rules for governance
- running in its own process



# Decision point


- Custom Code or Third party software ?  
Should we welcome third party software?
  - Third party dependencies can be controlled as well
  - Standards are important
  - The core business domain is what we are good at
  - Everything else is someone else's expertise area
  - There is a library or a tool for everything
  - Speed up development







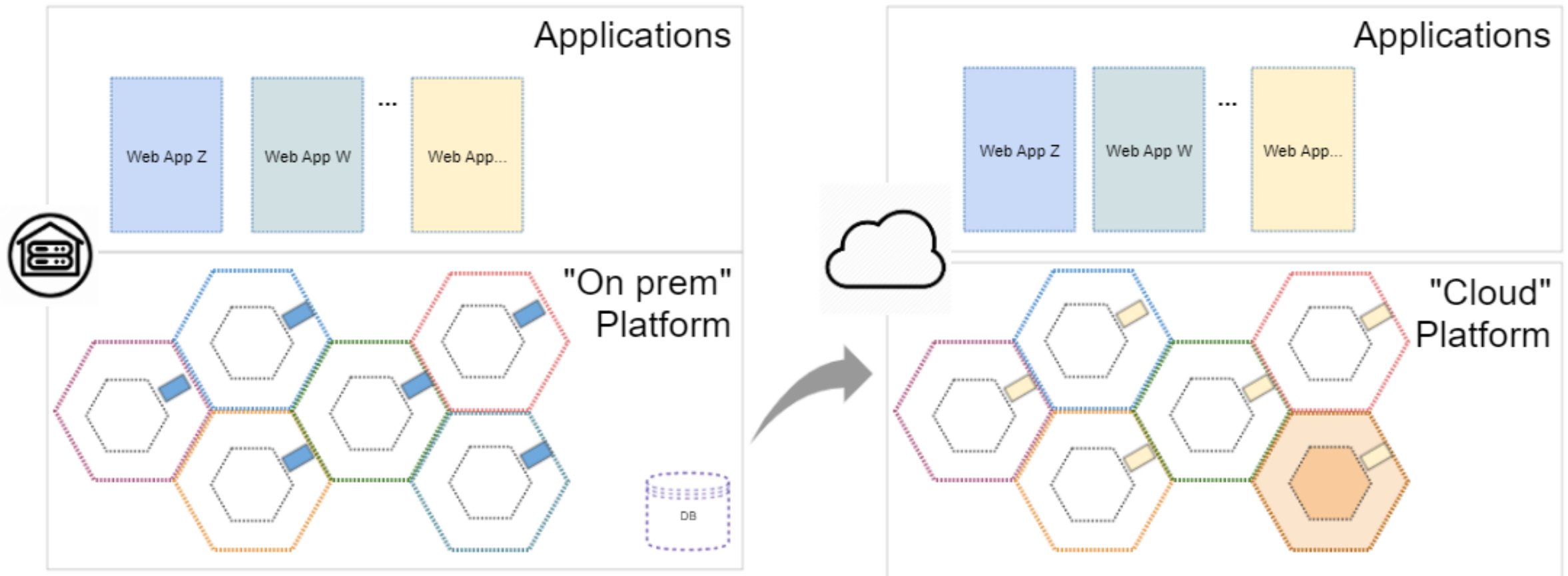
# Decision point

- **Are we ready to be a SaaS company? Should we take a “cloud first” approach for new development?**
  - it was a tough business decision made with confidence that the software team will be able to deliver
  - paradigm shift of the organization, as it applies to a larger scope than just this product development
- 



# Next step in evolution – Cloud & On premise

- “dual” applications
- 2 Platform implementations: Cloud and “On premise”



# Current challenges

- Security
- Complexity of tools like Kubernetes
- Grow skills inside development teams
- Regulatory compliance seems a topic in early ages in cloud
- Connectivity with the analytical instruments in the lab

This is the path for today, and new learning will come...

# Summary



2016

- Rearchitecture decision



2018

- Application segregation
- Rewrite start

2020

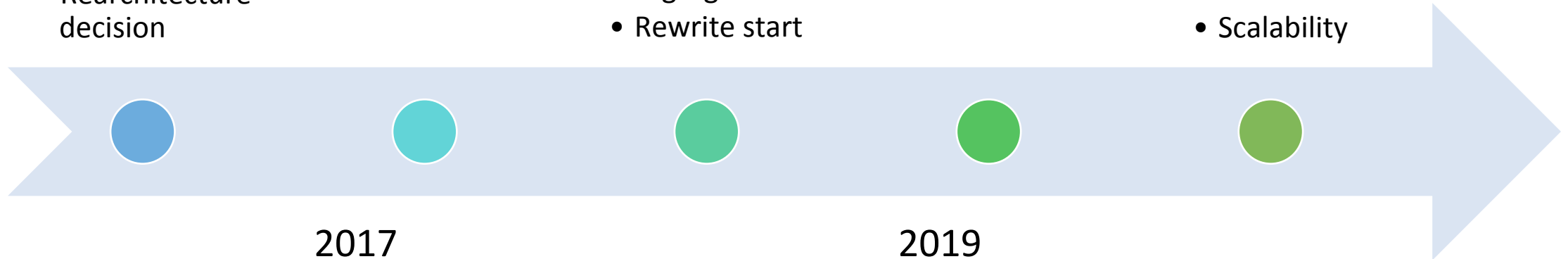
- Cloud
- Scalability

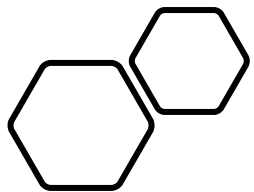
2017

- Layers creation

2019

- Platform & applications
- Quarterly release





# Personal retrospective



The experience really helped me to grow as software architect.



Decisions could save or add months of work on development teams.



Satisfaction is huge when you see that what you designed is actually working.

The background is a solid orange color. It features several yellow geometric elements: a large solid circle in the upper left, a smaller solid circle in the top right, a dashed vertical line in the top left, a dashed diagonal line in the bottom left, a solid square outline on the left, and a solid line forming a corner or 'L' shape in the top center.

Questions?