accenture

MICROSERVICE ARCHITECTURE IN ACTION

IOANNIS KORMARIS
FINANCIAL SERVICES

SEPTEMBER 2018

"Failure is simply the opportunity to begin again, this time more intelligently." -- Henry Ford

AGENDA

Architecture

Scaling

Service 2 Service

Deployment

Monitoring

ARCHITECTURE

“The fundamental organization of a system, embodied in its components, their relationships each other and the environment, and the principles governing its design and evolution.”

ANSI/IEEE Stf 1471 – 2000

SOFTWARE ARCHITECTURE

Architecture is NOT beyond programming

Most critical components

Expert developers' shared understanding of the system design

The set of design decisions that must be made early

The decisions that you wish you could get right early

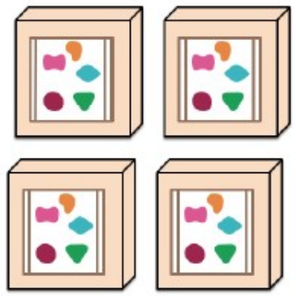
The decisions that are hard to change

TRADITIONAL VS MICROSERVICES ARCH

Traditional architecture

A Monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers



UI specialists

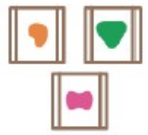
Middleware specialists

DBAs

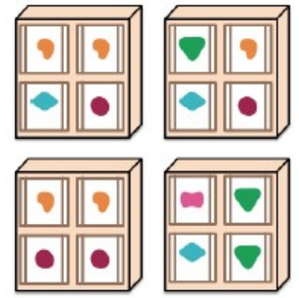
Siloed functional teams aligned around technology layers

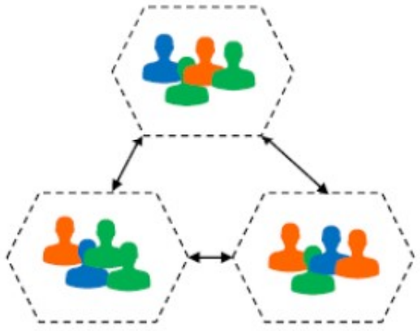
Microservices architecture

Microservices architecture puts each element of functionally into a separate service...



... and scales by distributing these services across servers, replicating as needed.





Cross functional teams aligned around business lines

WHY?

MICROSERVICES PRINCIPLES

Organize around business capabilities

Scalability

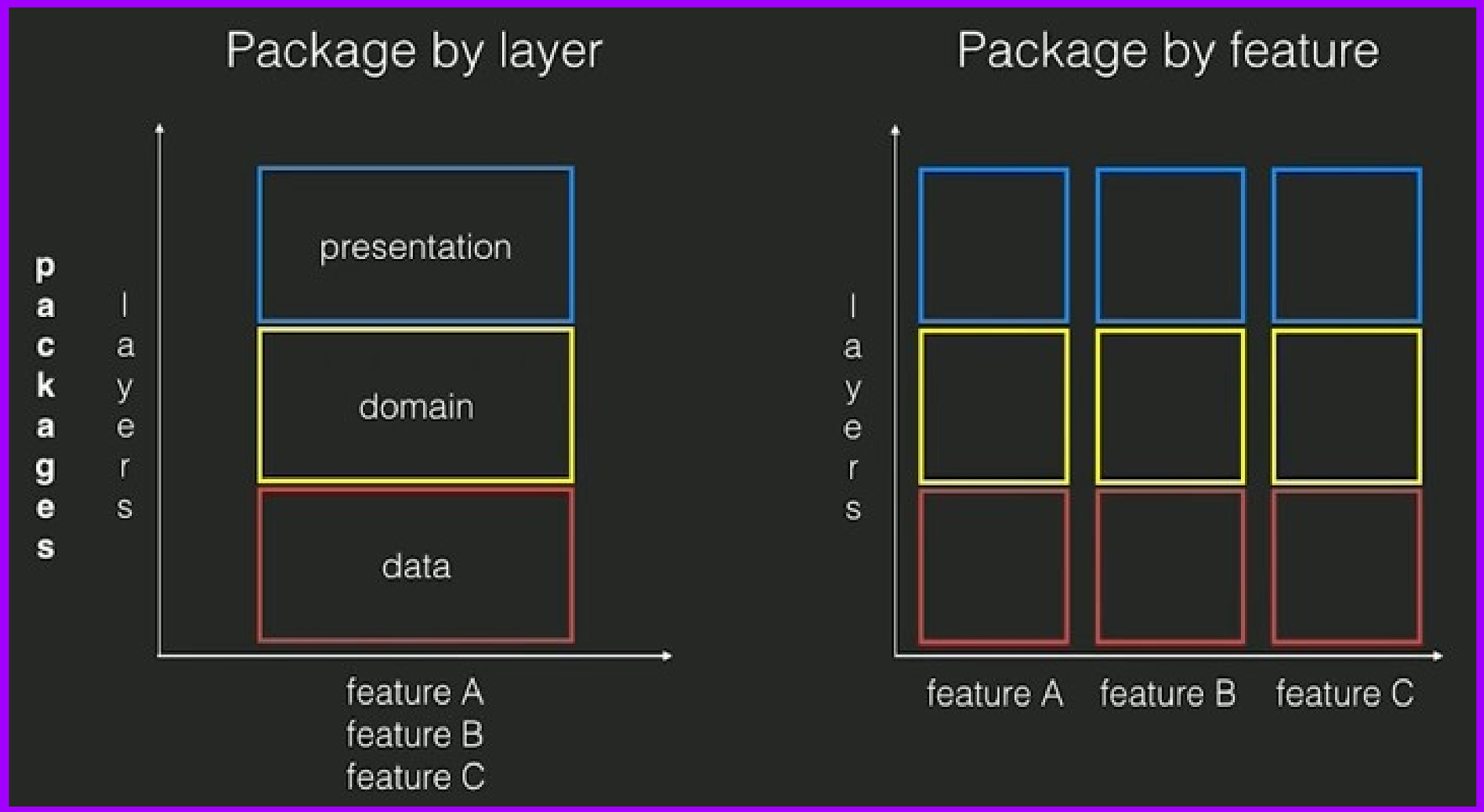
Target outcomes, not projects

Domain Specific

DOES THIS RING A BELL ALREADY ?

Package By Layer vs Package By Feature

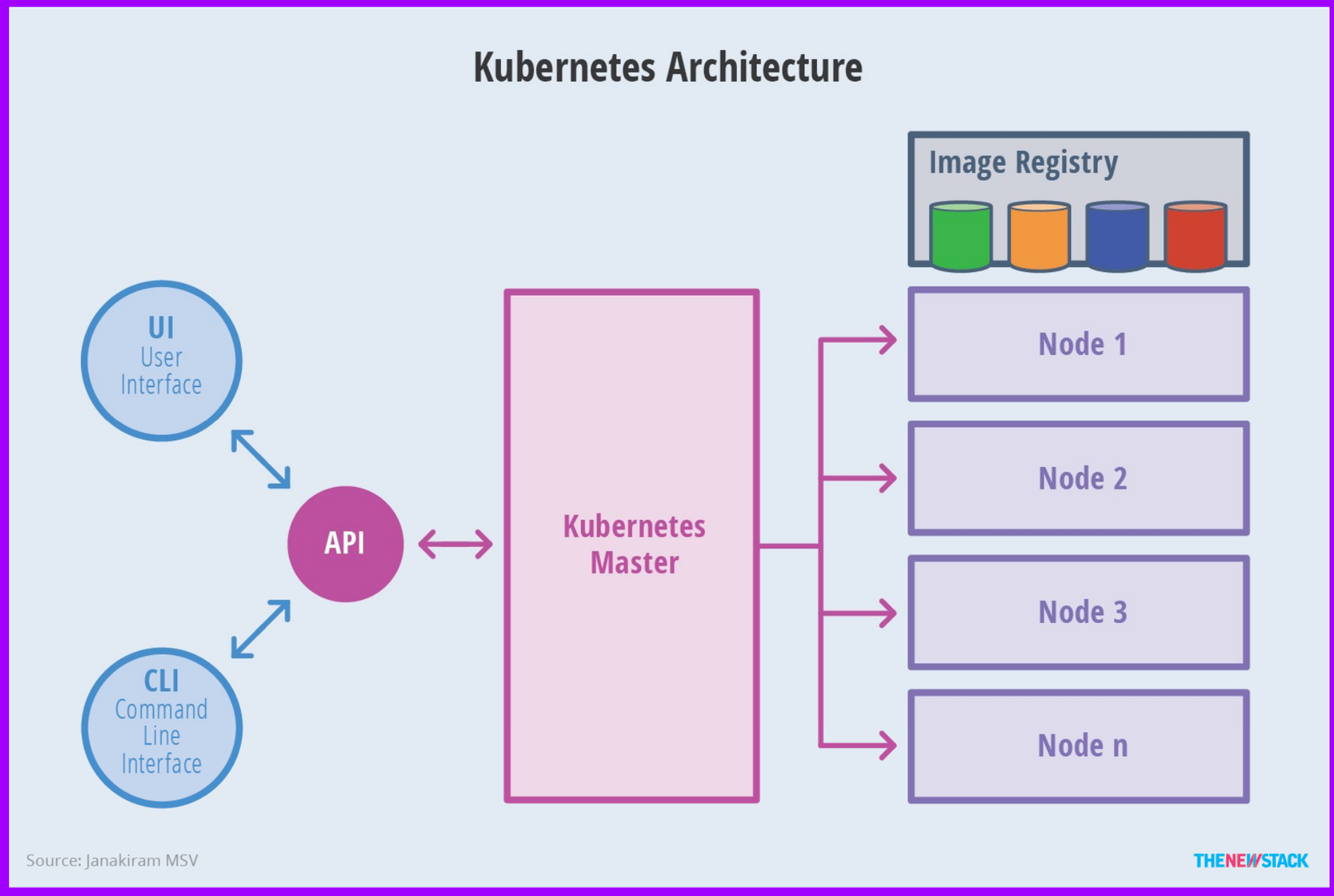
PACKAGE BY LAYER VS PACKAGE BY FEATURE



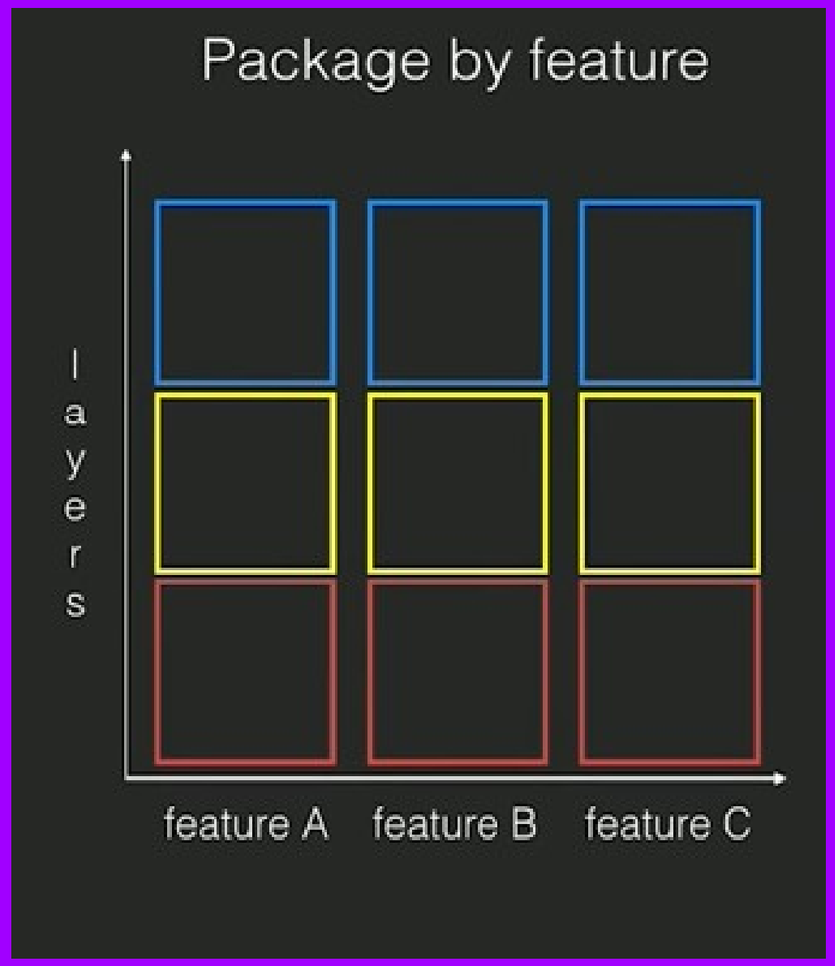
I WANT A PLATFORM TO DEPLOY

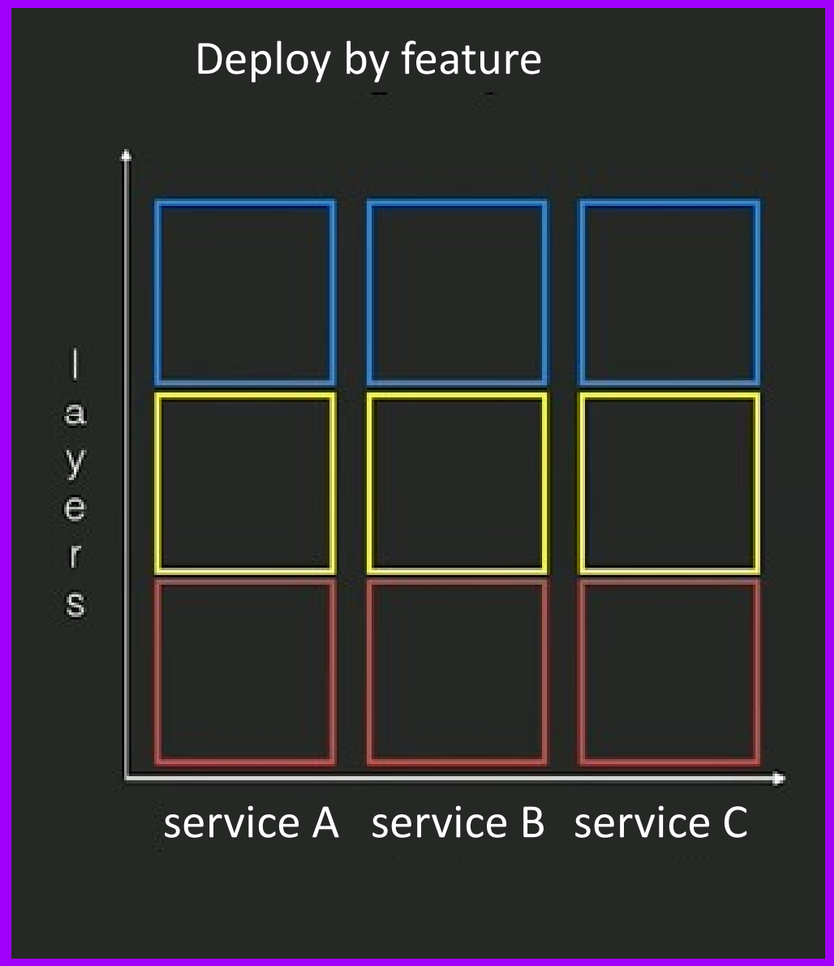
FEATURES

KUBERNETES


Kubernetes Architecture
Image Registry
UI
User
Interface
API
CLI
Command
Line
Interface
Kubernetes
Master
Node 1
Node 2
Node 3
Node n
Source: Janakiram MSV
THENEWSTACK

PACKAGE BY FEATURE WILL BECOME MICROSERVICE





HOW MANY FEATURES PER MICROSERVICE ?

I DON’T KNOW

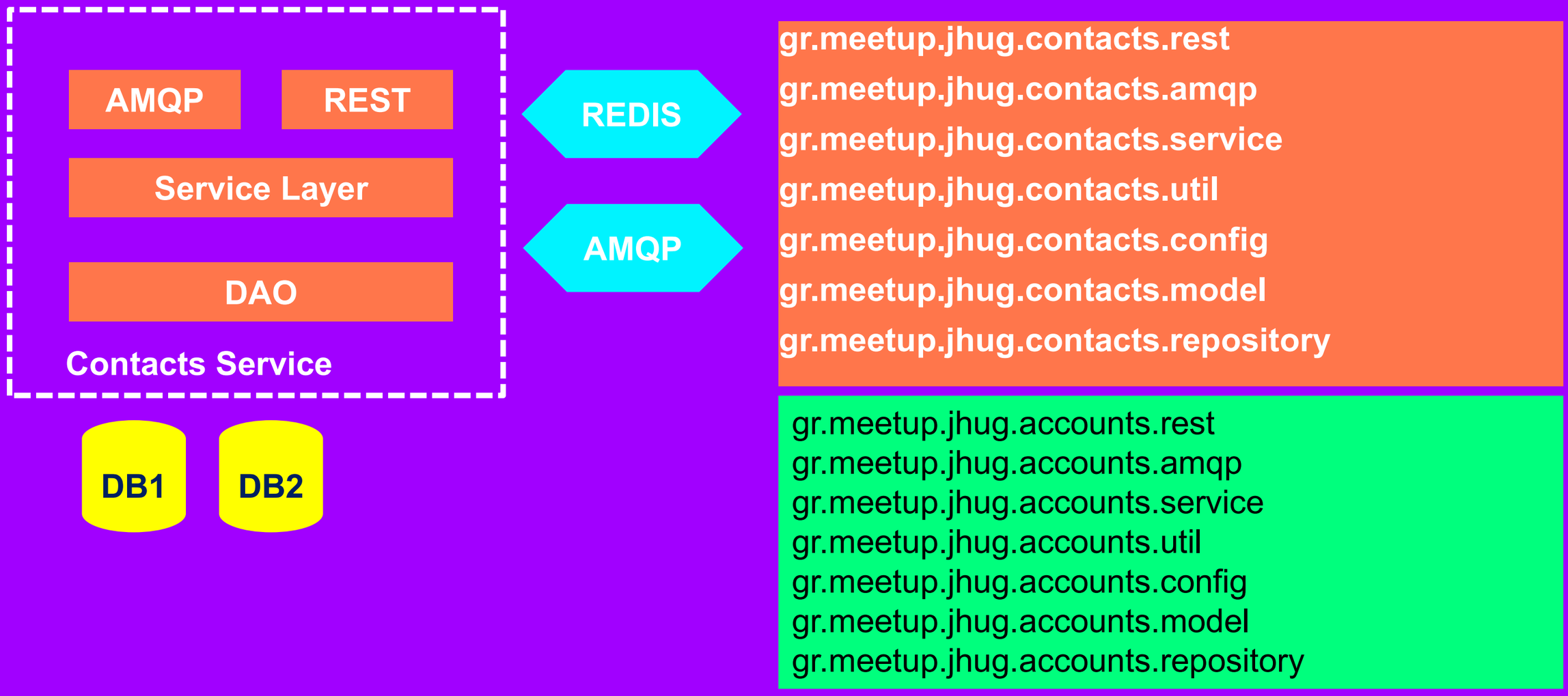
AND NOW WHAT... ?

GUESS

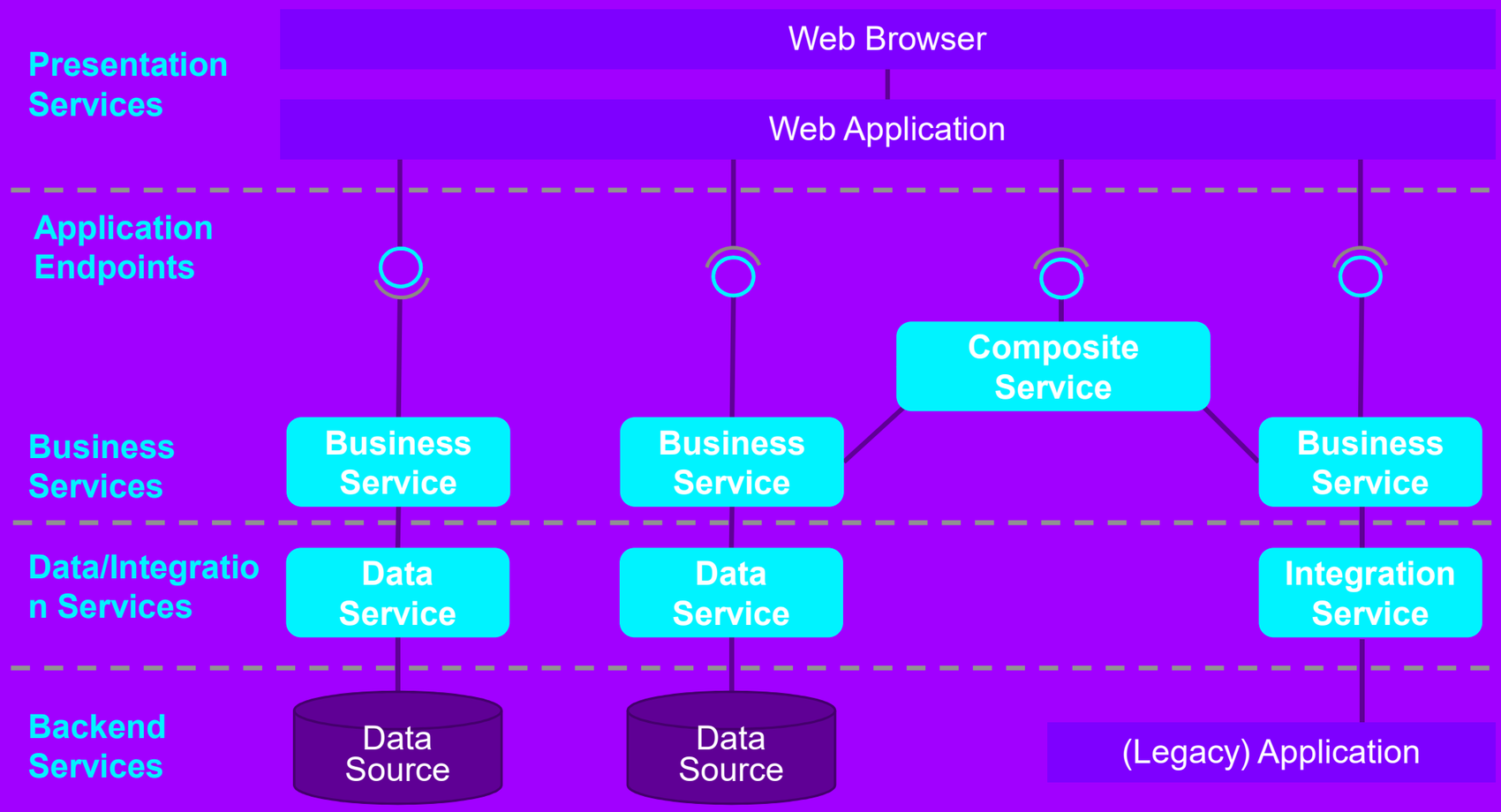
HOW DO I DESIGN MY SERVICE ?

Package by FEATURE

PACKAGING EXAMPLE



COMPLEX DOMAIN



HOW MANY DATABASES ?

Deploy ONE Database Instance:

- **Easy to maintain**
- **Easy to monitor**
- **Easy to backup**

Separate Microservices Domains either by schema/database

- **Vendor specific**
- **Domain Isolation**

SCALING

ONE SERVICE MANY INSTANCES

Accounts Service

Accounts Service

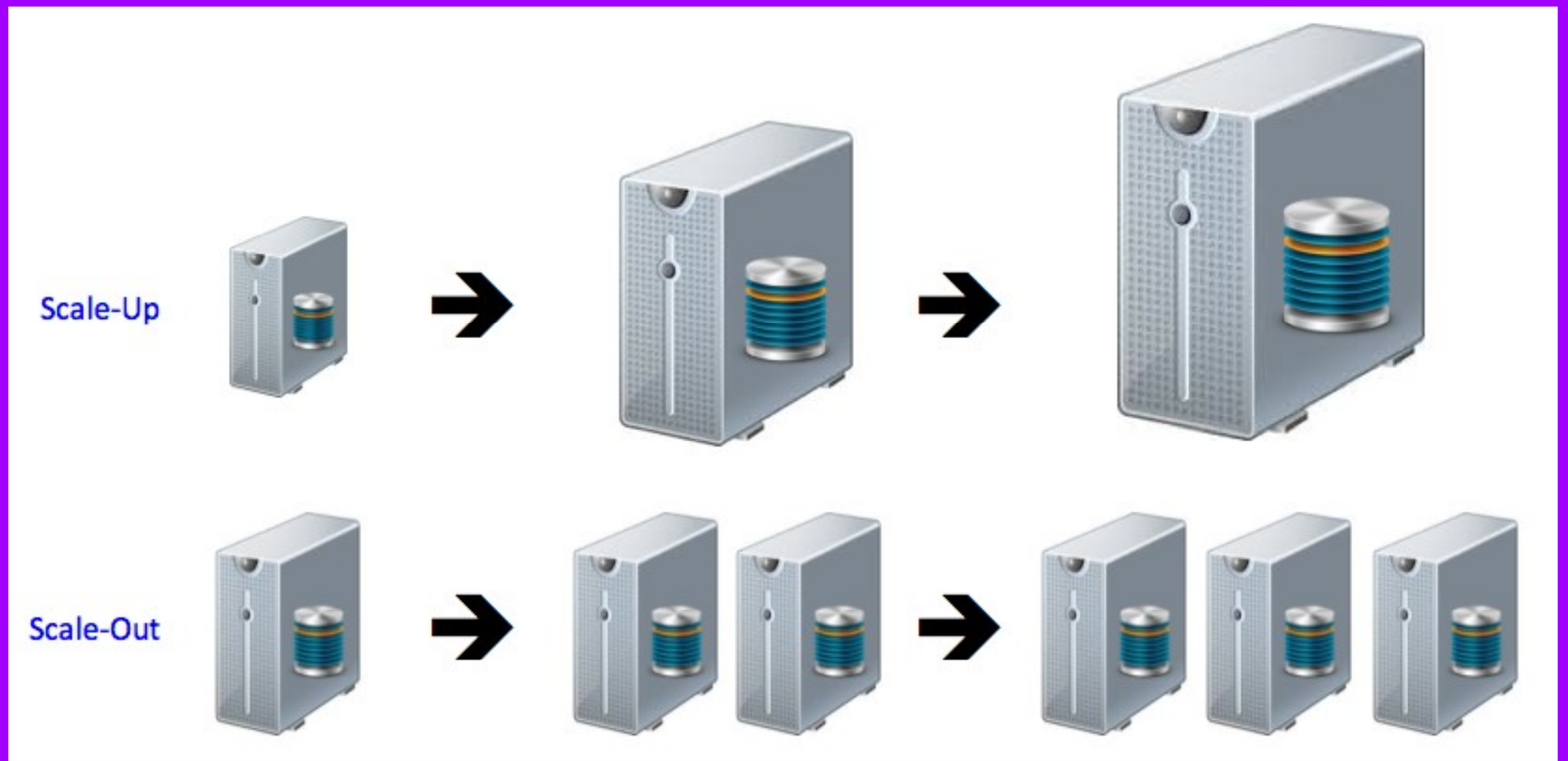
Accounts Service

Accounts Service

Accounts Service

Accounts Service

SCALE UP VS SCALE OUT



SCALE UP VS SCALE OUT

Scale Up

Increase CPU/MEM per container

Increase threads

Reach maximum performance per container

Attention Thread Safety

In process shared resources

Scale Out

Increase processes (i.e. spawn containers/replicas)

Attention Process Safety

External Shared Resources (e.x. DB, Distributed Cache - Redis)

ALWAYS MIND SHARED RESOURCES

SERVICE 2 SERVICE

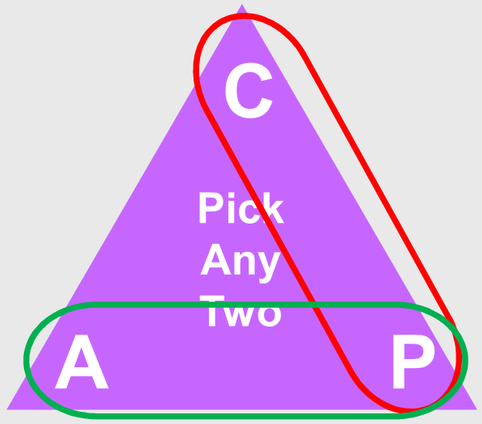
COMPLEXITY

Microservices force a move towards distributed computing, and there are some basic things about distributed computing that we must always keep in mind.

Fallacies of distributed computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

CAP Theorem – Pick Any Two

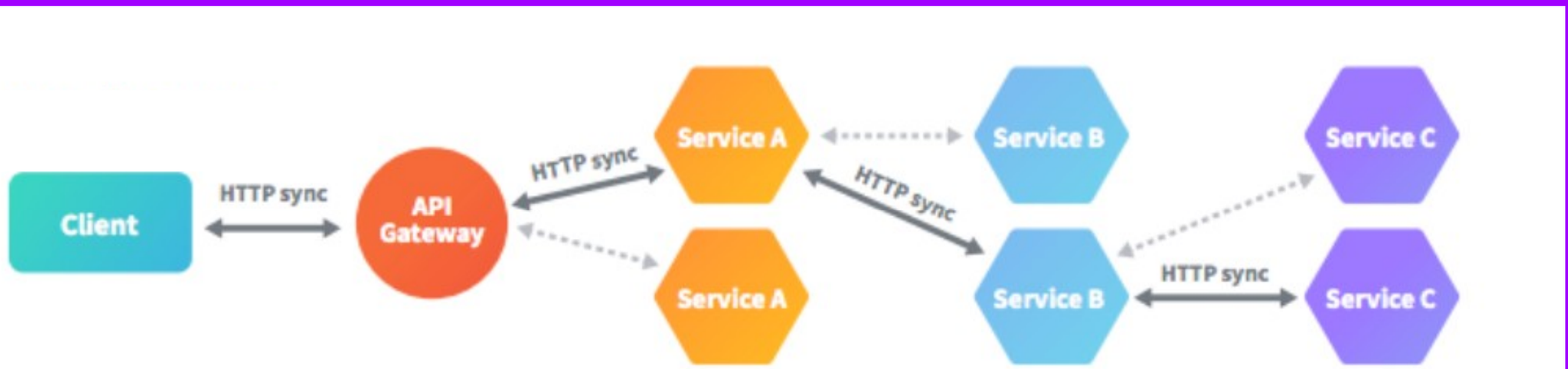


Consistency
Each node shows the same data at all times

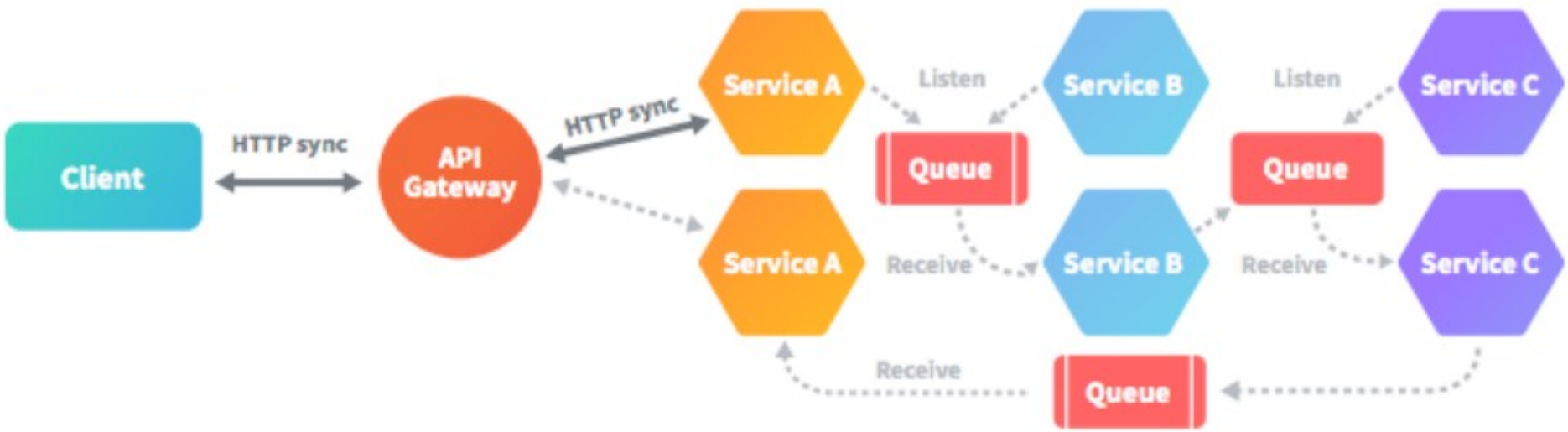
Availability
Each node is available for requests at all times

Partition Tolerance
Able to handle network outages

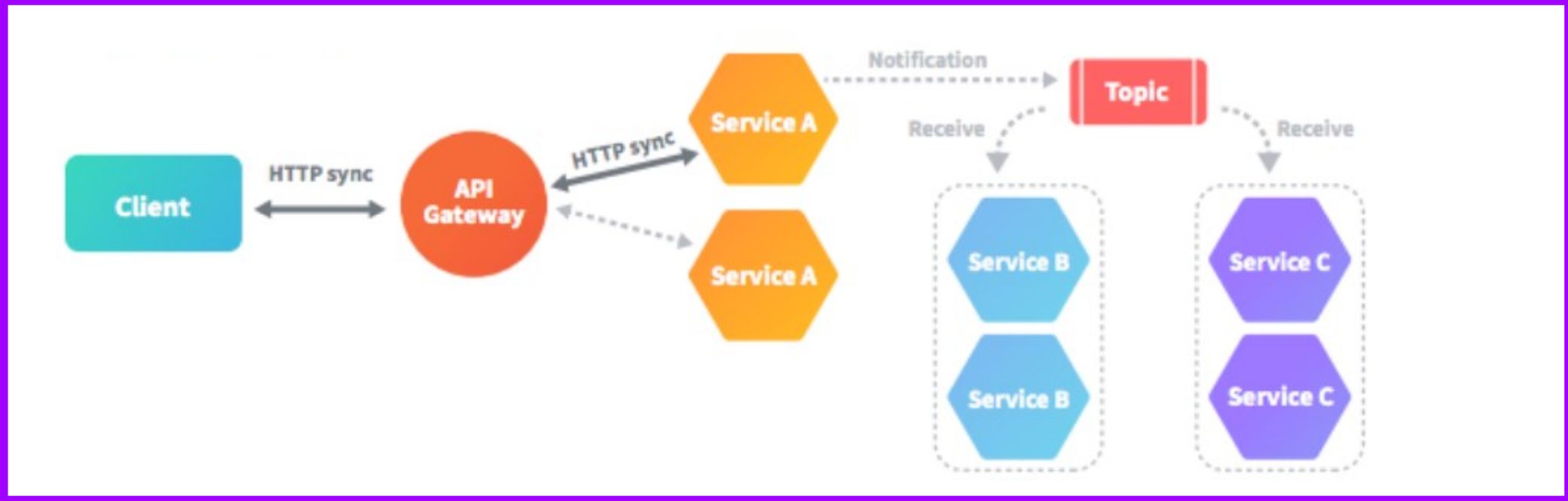
SYNCHRONOUS



ASYNCHRONOUS



PUBLISH SUBSCRIBE



REST VS AMQP

Rest	AMQP
Rest API	RPC Support
Synchronous by Nature	Asynchronous by Nature
API documentation (OpenAPI)	Hard to Document
MS deployed as Kube Service	MS is AMQP Client

FUTURE CONCERNS

Service Mesh (e.x. Istio)

Reactive programming

DEPLOYMENTS

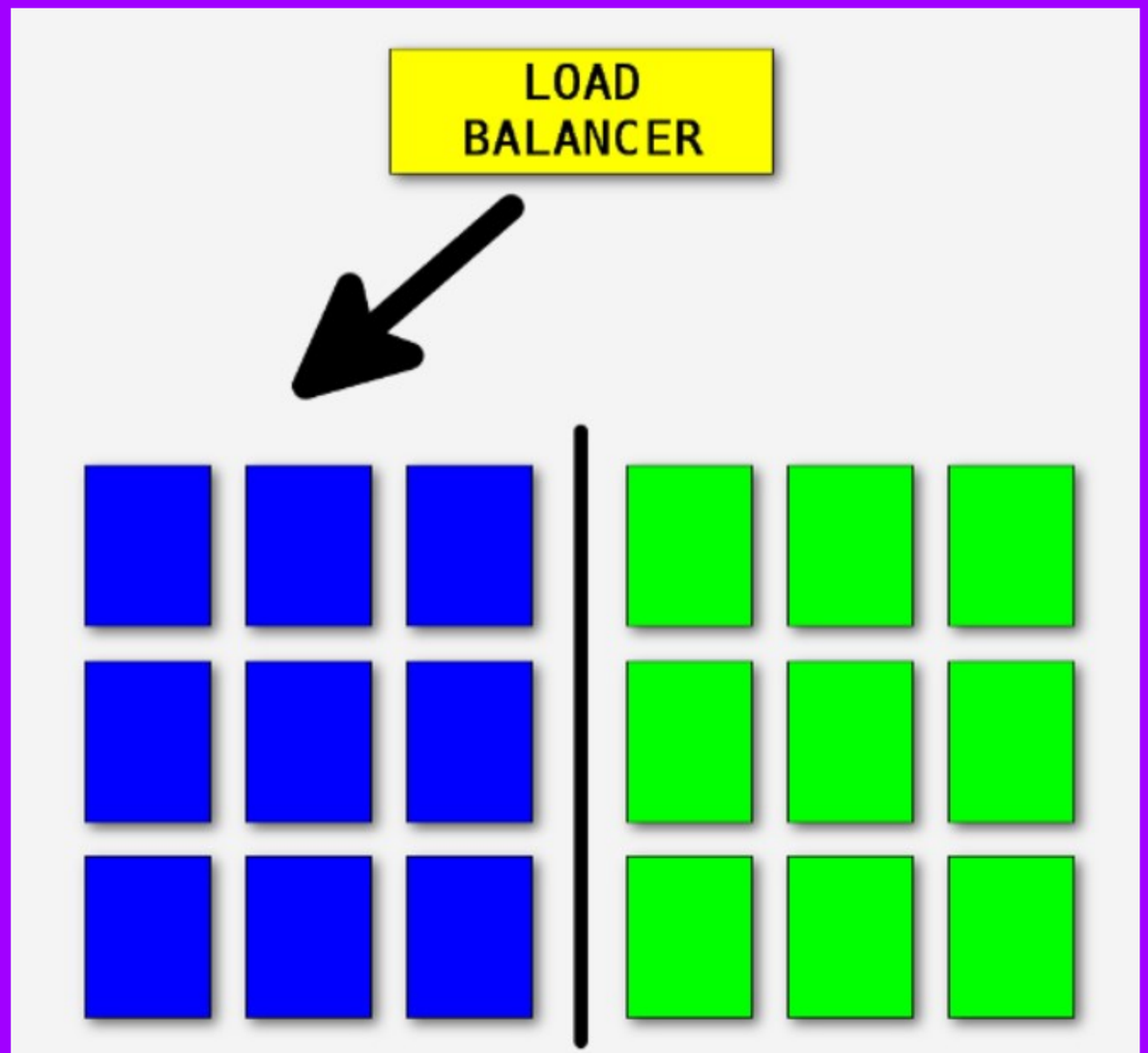
COLORFUL-DEPLOYMENTS

BLUE GREEN

CANARY

ROLLING

BLUE GREEN

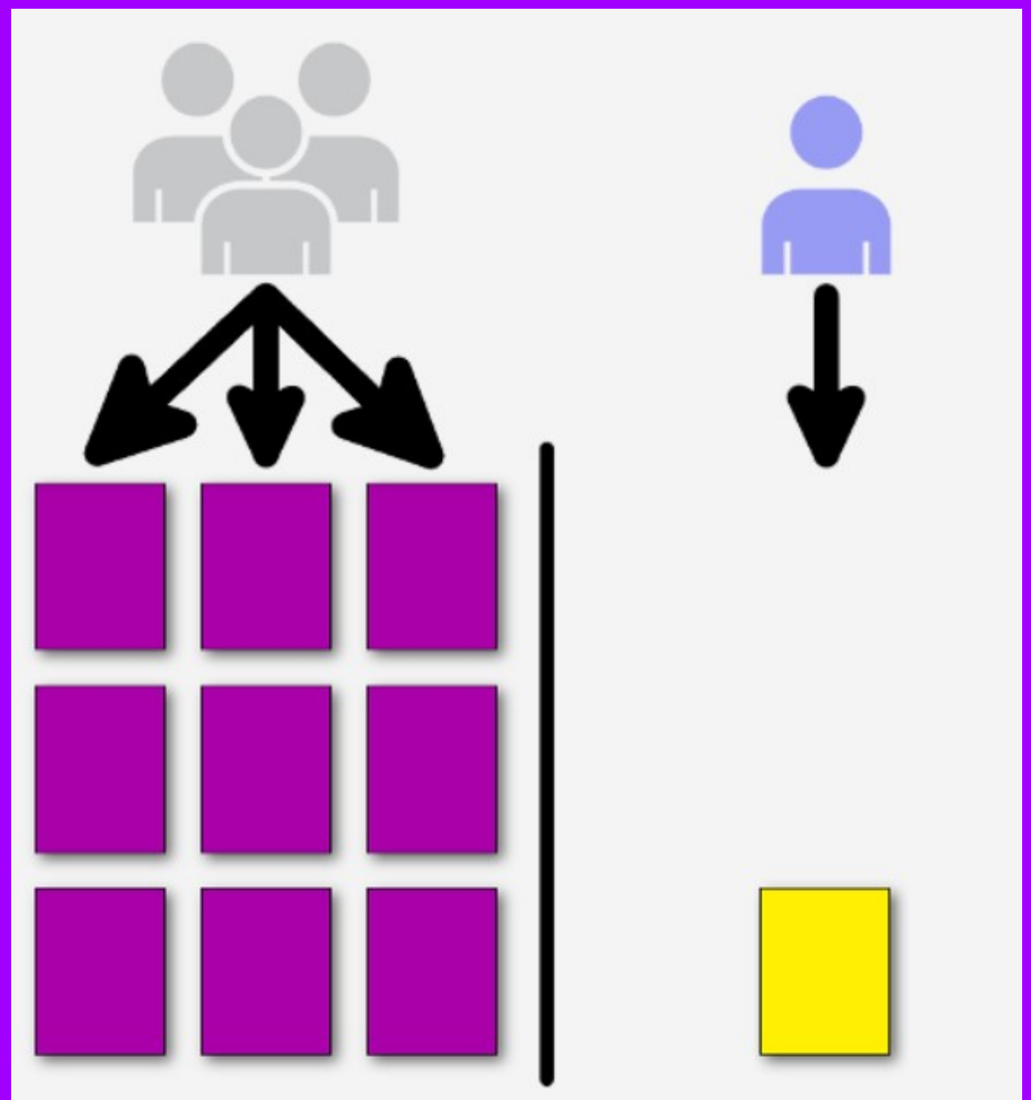


Redirect traffic when green is ready

Define what ready is

Mind Shared Resources

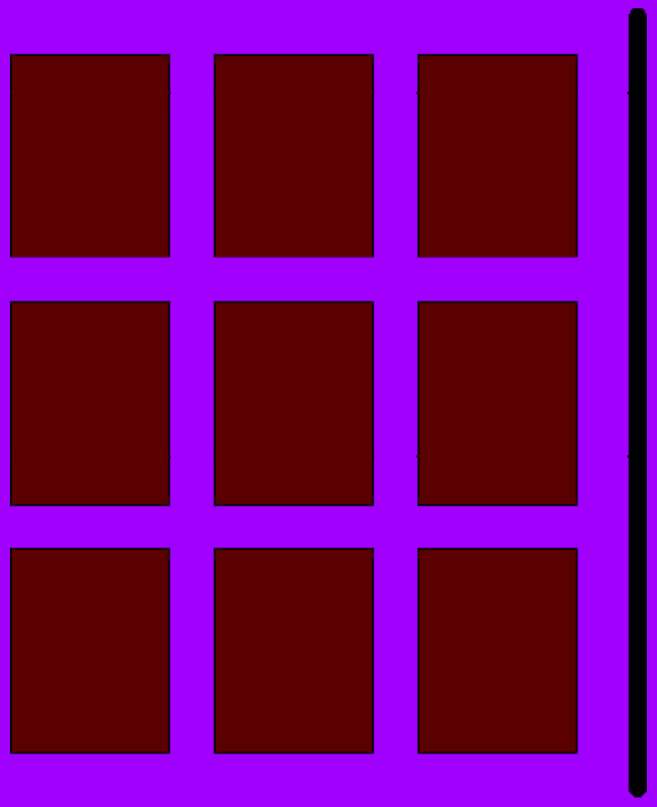
CANARY



Feature toggle

GateKeep @ Facebook

ROLLING DEPLOYMENT



Slowly replaces currently running instances of our application with newer ones.

READINESS AND LIVENESS

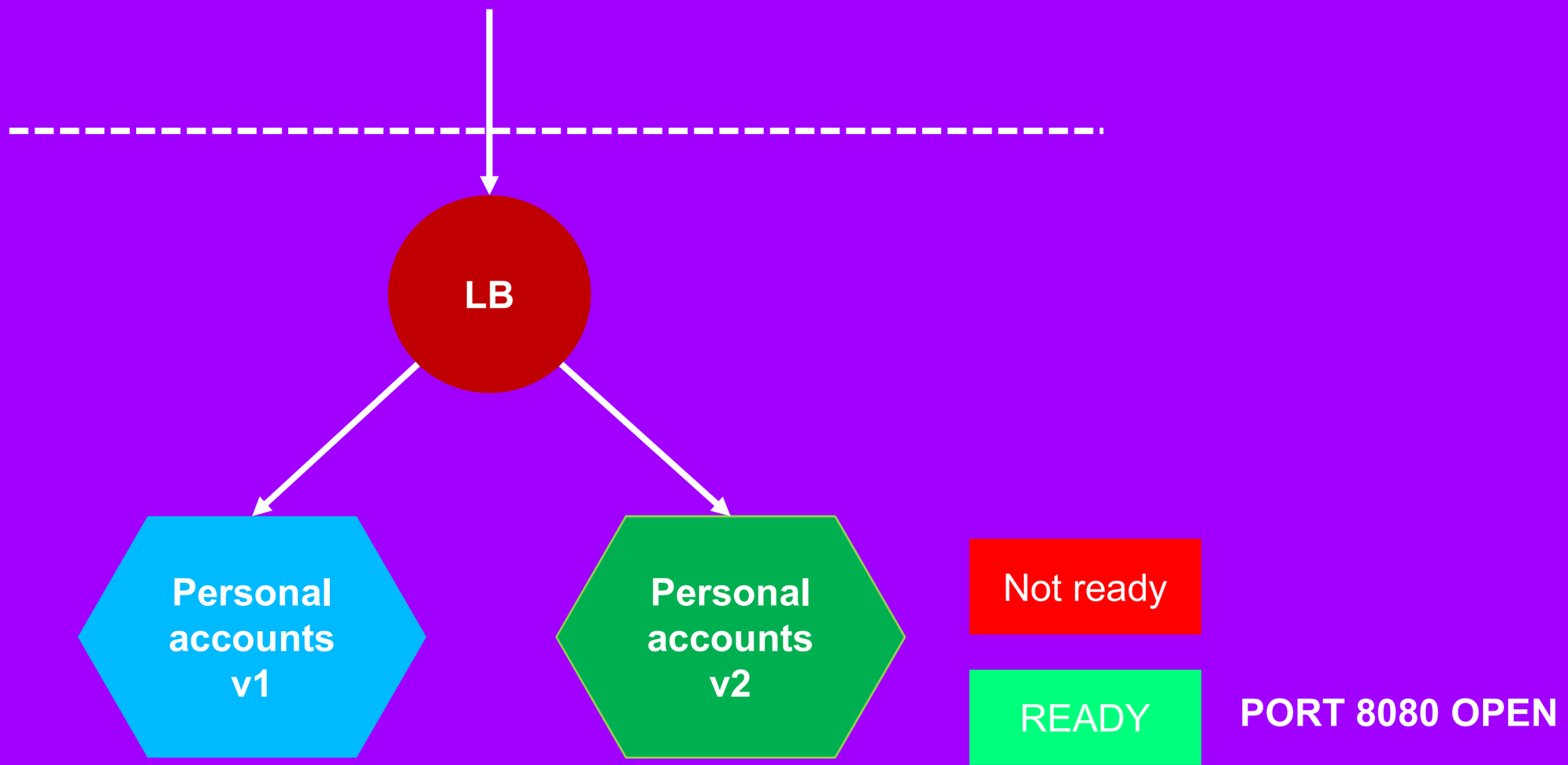
Readiness Probe

A readiness probe determines if a container is ready to service requests.

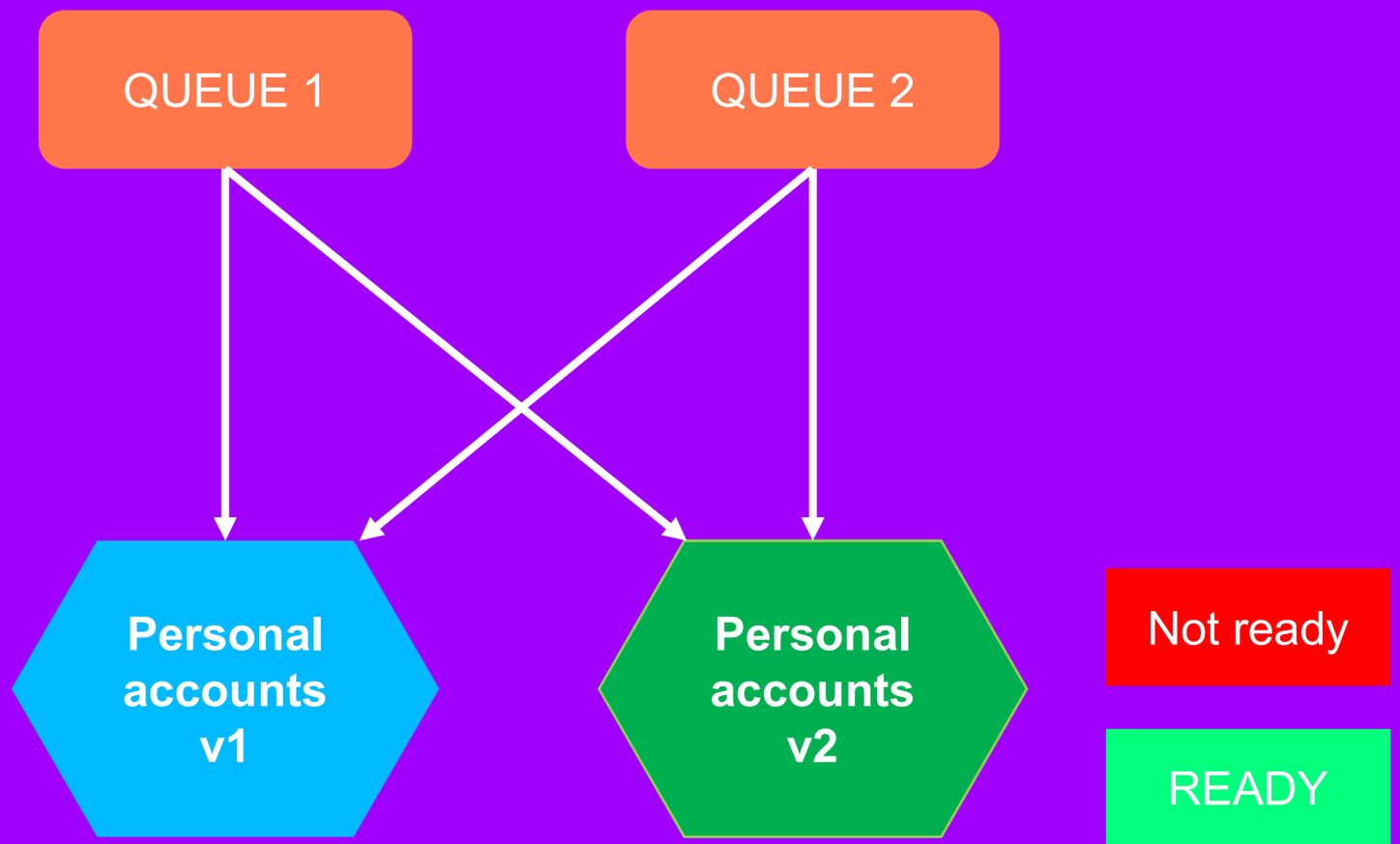
Liveness Probe

A liveness probe checks if the container in which it is configured is still running.

READINESS EXAMPLE REST

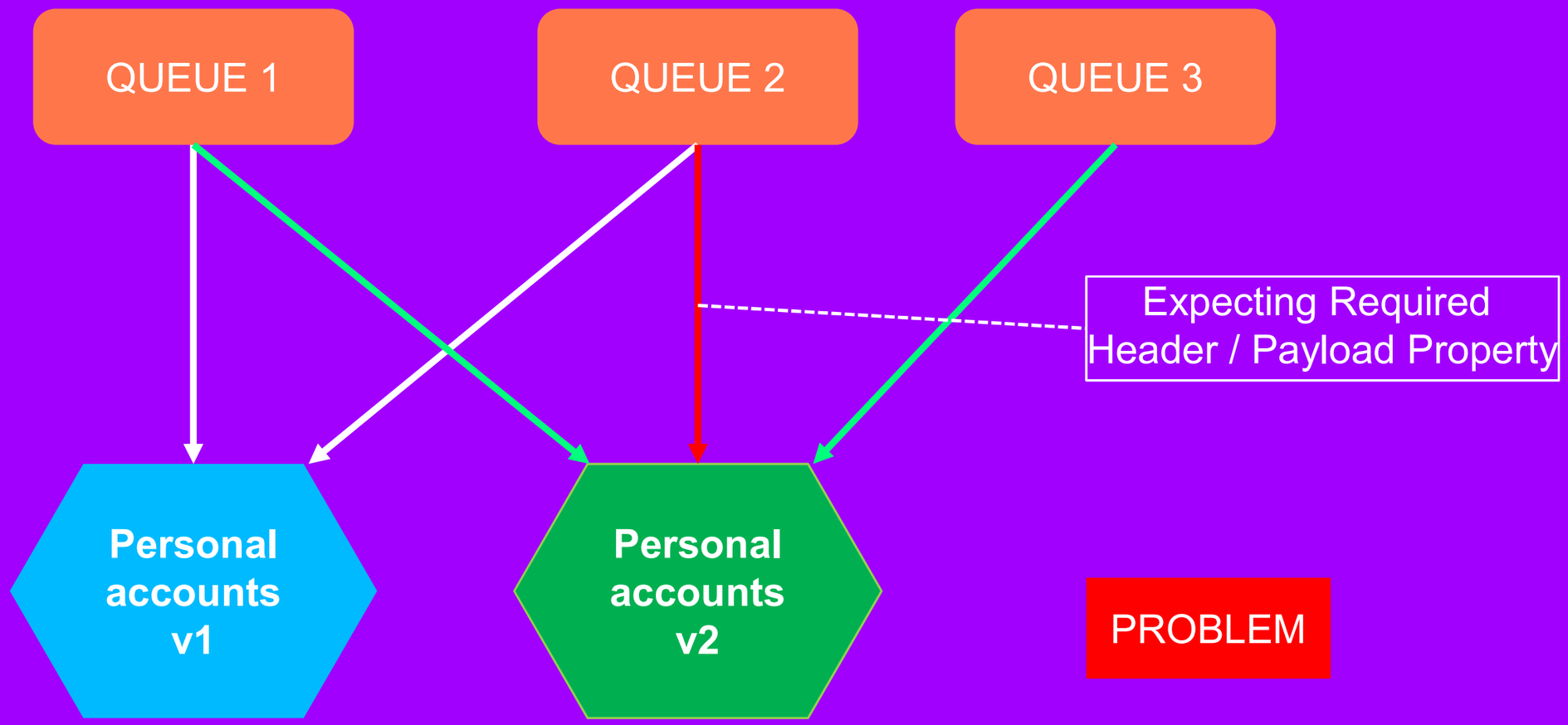


READINESS EXAMPLE AMQP



PORT 8080 MAY BE OPEN

ROLLING DEPLOYMENT DEPENDENCIES



READINESS IMPLEMENTATION HINTS

REST INTEGRATION

Spring Boot Actuator endpoint /health

AMQP Integration

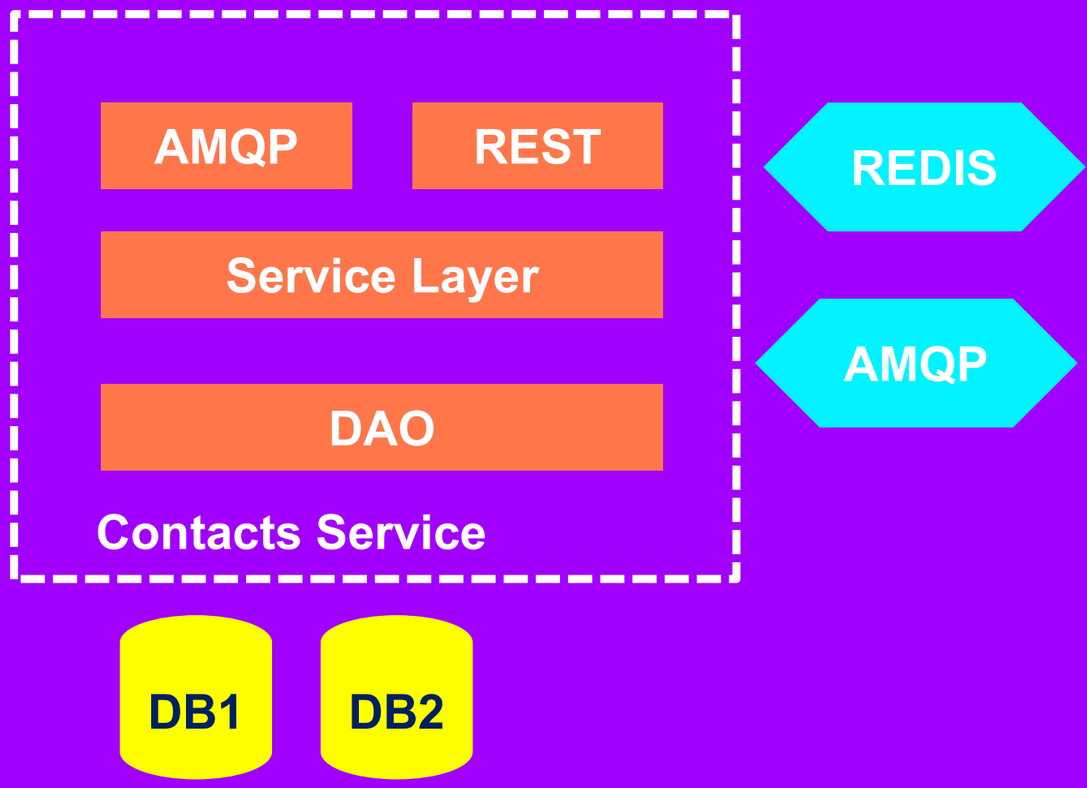
You may need to implement your own healthcheck

Application needs to starts consuming on all Queues

HINT : org.springframework.amqp.rabbit.listener.AsyncConsumerStartedEvent

Graceful shutdown

MY SERVICE IS READY WHEN



1. Has connection with Redis
2. Has connection with AMQP
3. Has connection with DB1
4. Has started consuming in all queues (n/n)

Maybe DB2 is optional (code smell)

EXIT WITH GRACE

Don’t just die

Handle SIGTERM

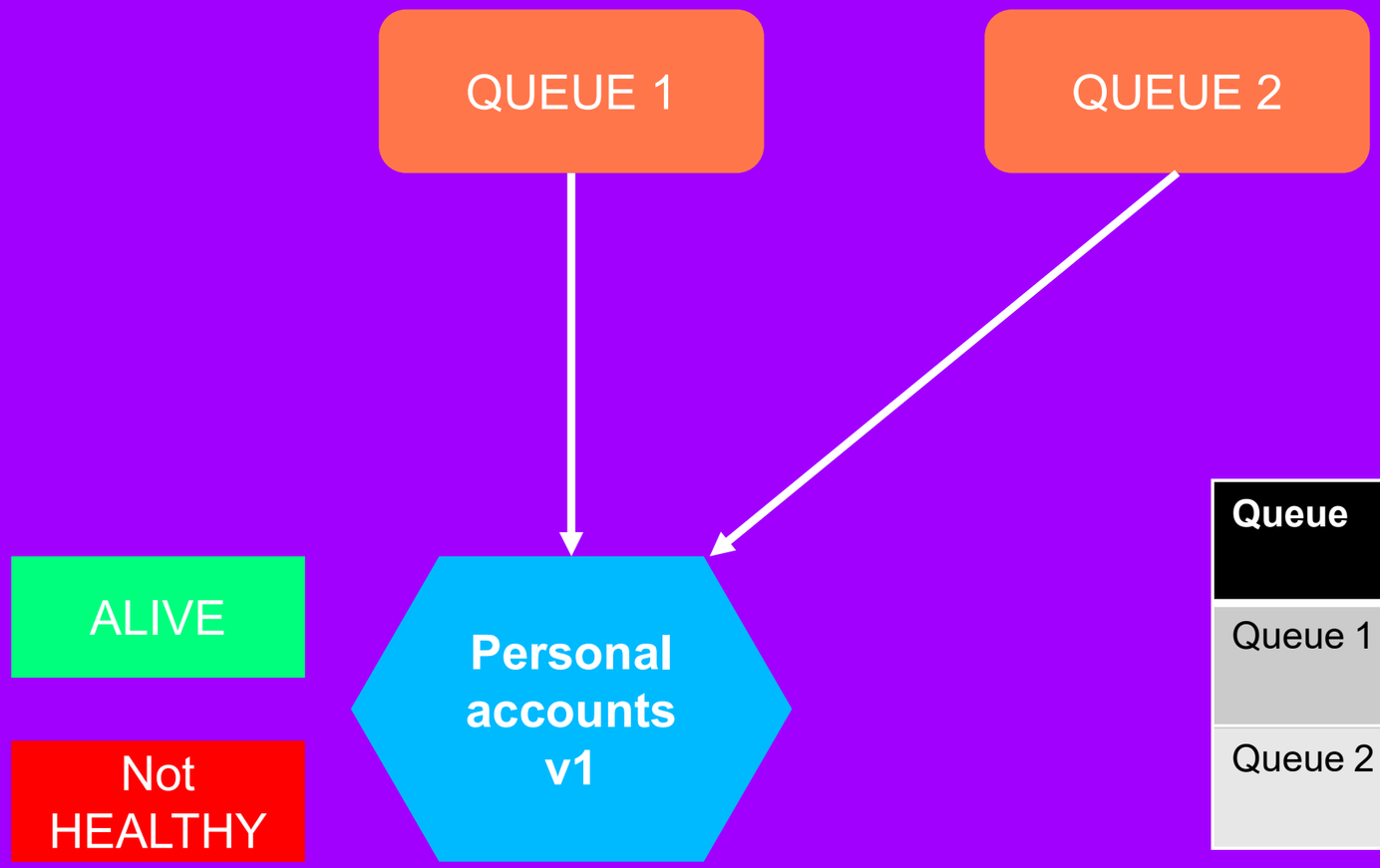
Container Process Running in PID 1

Commit Transactions

Release Shared Locks

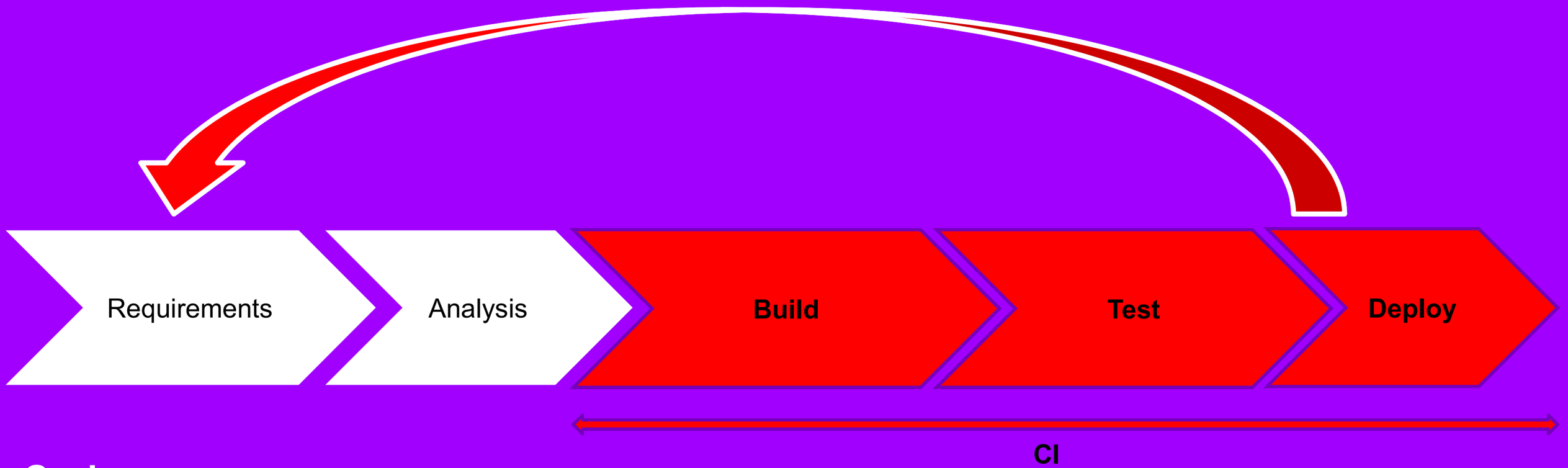
Kubernetes will kill you with SIGKILL under certain conditions

LIVENESS EXAMPLE AMQP



Queue	Consumers	Ready	Unacked	Total	Ack Rate
Queue 1	2	4	2	6	0 m/s
Queue 2	2	4	2	6	2 m/s

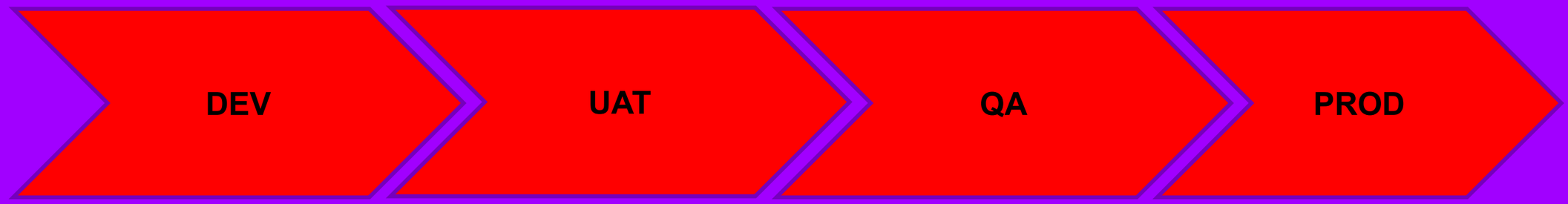
CONTINUOUS INTEGRATION



Goals

- Reduce Risks/Minimize Costs
- Frequent Releases (2 week Sprints)
- Fail Early
- Code Quality

CONTINUOUS DEPLOYMENT



Challenges

- Promote Builds between environments
- Seamless Update – Zero Downtime
- Build Once – Deploy Everywhere
- Restrict Application Configuration as close to the Environment as it gets
- Environment Provisioning

WAIT

WHAT DO YOU DEPLOY ?

PRODUCT OR SERVICES

BoM (bill of Materials) of Services vs Service

Always have a pipeline per Service

Release Hardening further includes:

- **Penetration test**
- **Stress test**
- **Regression test**
- **Resilience test**

Overview

Applications

Builds

Resources

Storage

Monitoring

authentication created 2 months ago

Start Pipeline

Source Repository: http://zuul.eurobank.efg.gr/bitbucket/scm/ocp/pipeline.git

Recent Runs

Average Duration: 10m 21s

Build #12	Initialize	Checkout authentication ...	Run Unit Tests	Build JAR	Build Image v1.12.0	Deploy Image Repository ...	Deploy UAT & QA
8 minutes ago	0s	7s	27s	20s	37s	1m 28s	58s
View Log							

View Pipeline Runs | Edit Pipeline

authorization created 2 months ago

Start Pipeline

Source Repository: http://zuul.eurobank.efg.gr/bitbucket/scm/ocp/pipeline.git

Recent Runs

Average Duration: 8m 40s

Build #8	Initialize	Checkout authorization r...	Run Unit Tests	Build JAR	Build Image v1.13.0	Deploy Image Repository ...	Deploy UAT & QA
3 hours ago	0s	5s	28s	20s	36s	2m 11s	2s
View Log							

View Pipeline Runs | Edit Pipeline

batch-process created 2 months ago

Start Pipeline

Source Repository: http://zuul.eurobank.efg.gr/bitbucket/scm/ocp/pipeline.git

Recent Runs

Average Duration: 9m 4s

Build #8	Initialize	Checkout batch-process r...	Run Unit Tests	Build JAR	Build Image v1.11.0	Deploy Image Repository ...	Deploy UAT & QA
3 hours ago	0s	23s	1m 10s	21s	45s	3s	4s
View Log							

View Pipeline Runs | Edit Pipeline

campaign-mgt created 21 days ago

Start Pipeline

Source Repository: http://zuul.eurobank.efg.gr/bitbucket/scm/ocp/pipeline.git

Recent Runs

Average Duration: 2m 25s

Build #4	Initialize	Checkout campaign-mgt ...	Run Unit Tests	Build JAR	Build Image v1.10.0	Deploy Image Repository ...	Deploy UAT & QA
2 hours ago	0s	7s		15s	27s	3m 48s	7s
View Log							

Checkout campaign-mgt release/1.10.0

View Pipeline Runs | Edit Pipeline

cards created 2 months ago

Start Pipeline

DEPLOYMENT VS RELEASE

DEPLOYMENT: Deploy Code on Production.

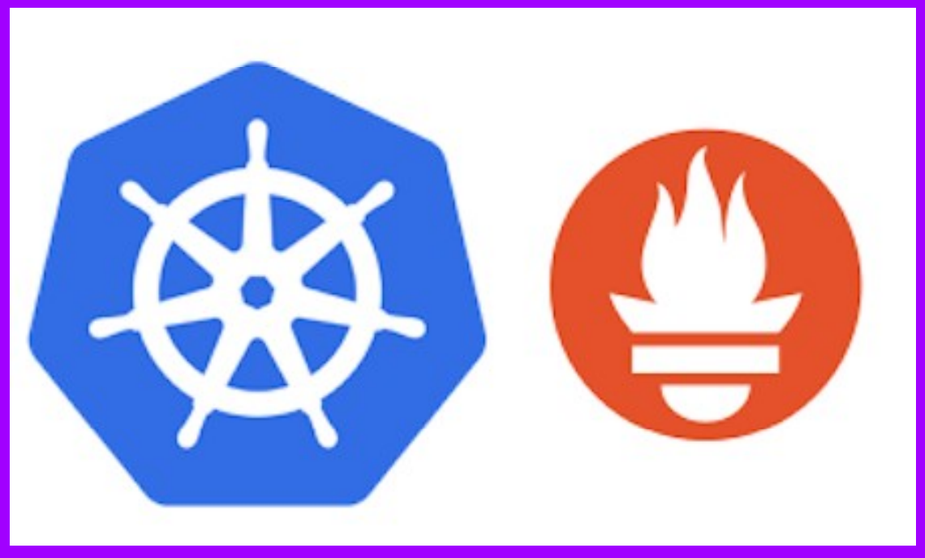
RELEASE: Feature available to Users. Deployment is a prerequisite.

MONITORING

COLLECT METRICS

Application Metrics

Infrastructure Metrics

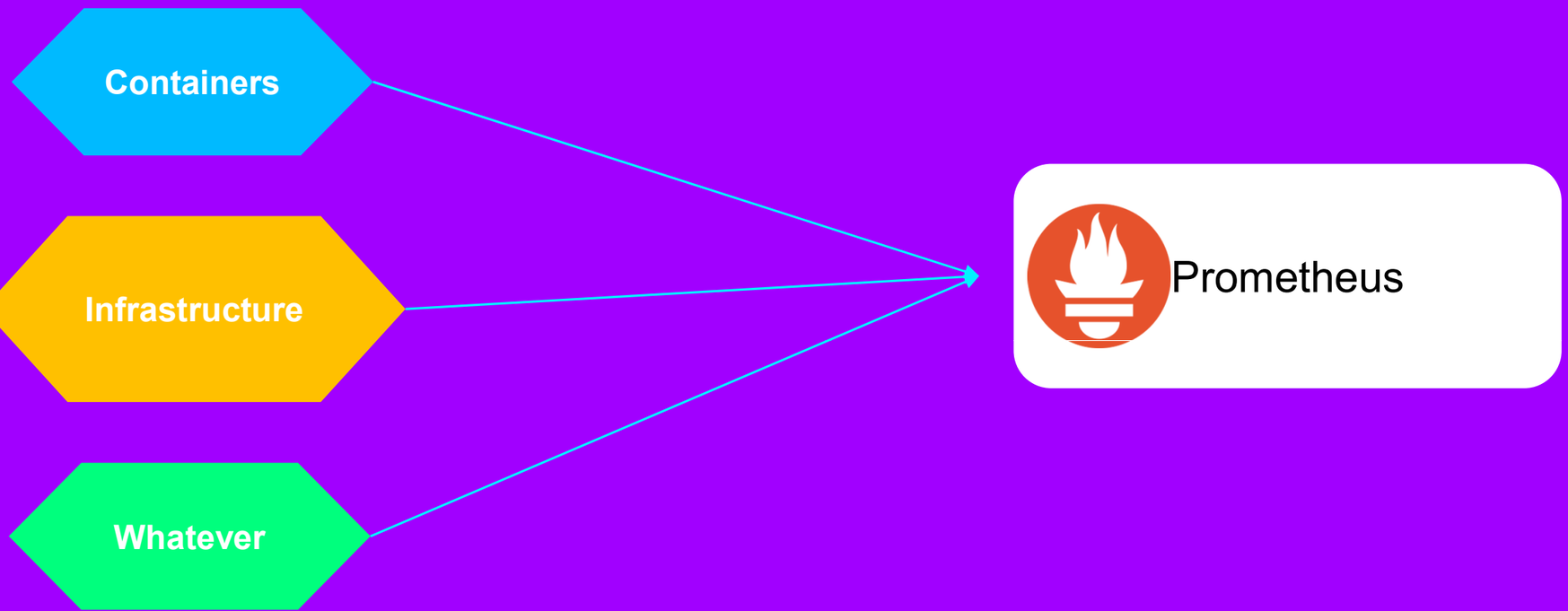


EVERYTHING

USEFUL METRICS

- Client & Server qps/errors/latency
- Every Log message should be a metric
- Every failure should be a metric
- Threadpool/queue size, in progress, latency
- Business logic inputs and outputs
- Data sizes in/out
- Process cpu/ram/language internal (e.x GC, Heap Size)
- Blackbox and end-to-end monitoring

PULL ARCHITECTURE



APPLICATION TIP

MicroMeter

Prometheus exporter for Spring Boot

Visualize in Grafana

REFERENCES

- https://12factor.net/
- https://martinfowler.com/articles/microservices.html
- https://dzone.com/articles/communicating-between-microservices
- https://docs.openshift.com/index.html
- https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html
- https://opensource.com/article/17/5/colorful-deployments
- https://micrometer.io/
- https://prometheus.io

accenture

THANK YOU