



Partnered with



MAKING OF YOUR APPLICATION CONTAINER READY

@TechProdevans

Deepak Mishra & Team

(CEO and Founder of Prodevans Technologies)

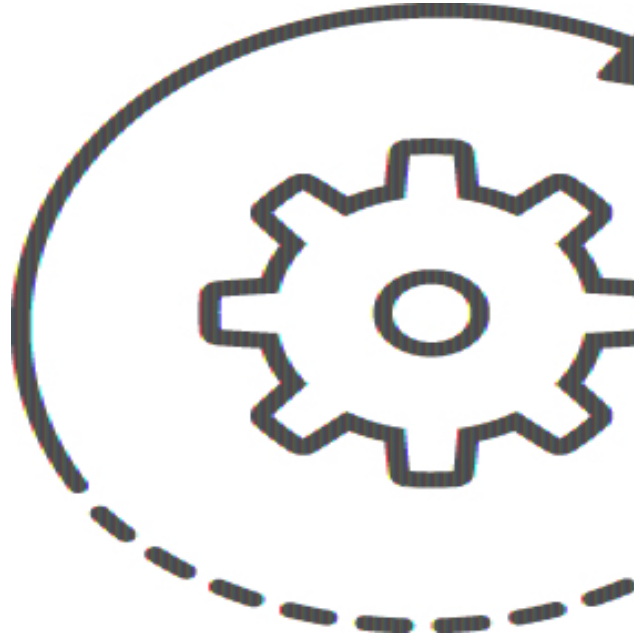
Chandrasahsa Vemu

(Senior DevOps Engineer, Solution Architect, Prodevans
Red Hat One Team)





Your Current State :-(



3 Month Deployment Cycle
3 Months BEFORE you gain Feedback and Learn



Your Journey to Microservices :-)



Re-Org to
DevOps



Self-Service,
On-Demand,
Elastic,
Infrastructure
as
Code
(Cloud)



Automation
Puppet, Chef,
Ansible
and/or
Kubernetes



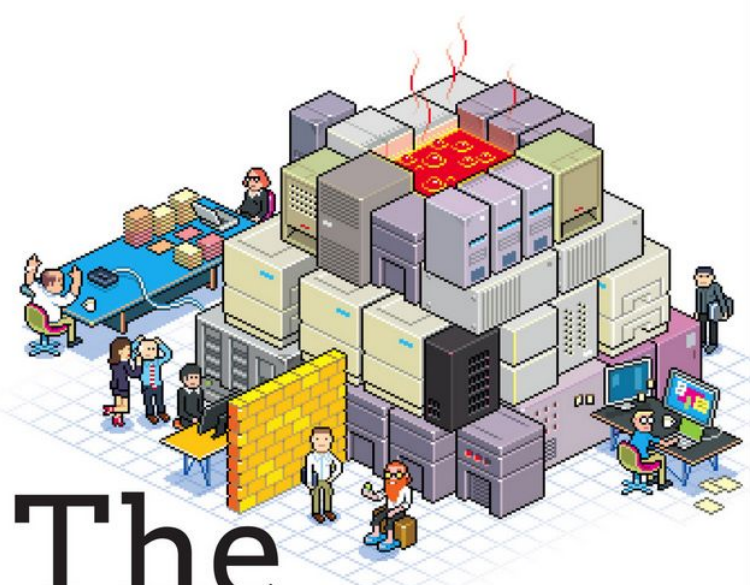
CI & CD
Deployment
Pipeline



Advanced
Deployment
Techniques



Silicon
Valley
DotCom
Startup



The Phoenix Project

A Novel About IT, DevOps, and Helping
Your Business Win

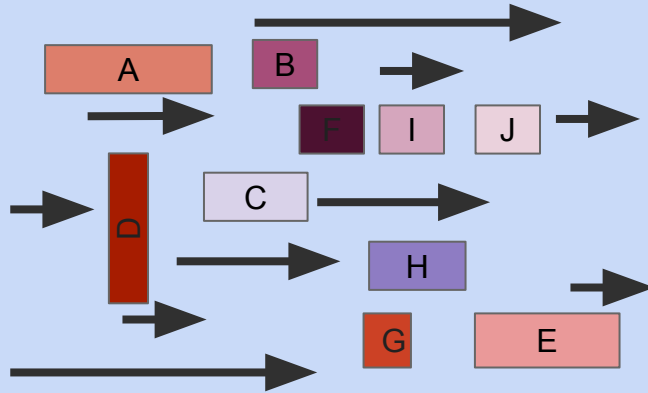
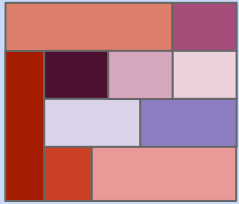
Gene Kim, Kevin Behr and George Spafford

10 Deploys a Day

How is that possible?

BUM - Break Up Monolith

10 Microservices



Deployment

Week 1: A, C, D

Week 2: A.1, H, G

Week 3: B, F, I, J, A.2

Week 4: A, C, D, E

Week 5: H, G, C.1, D.1

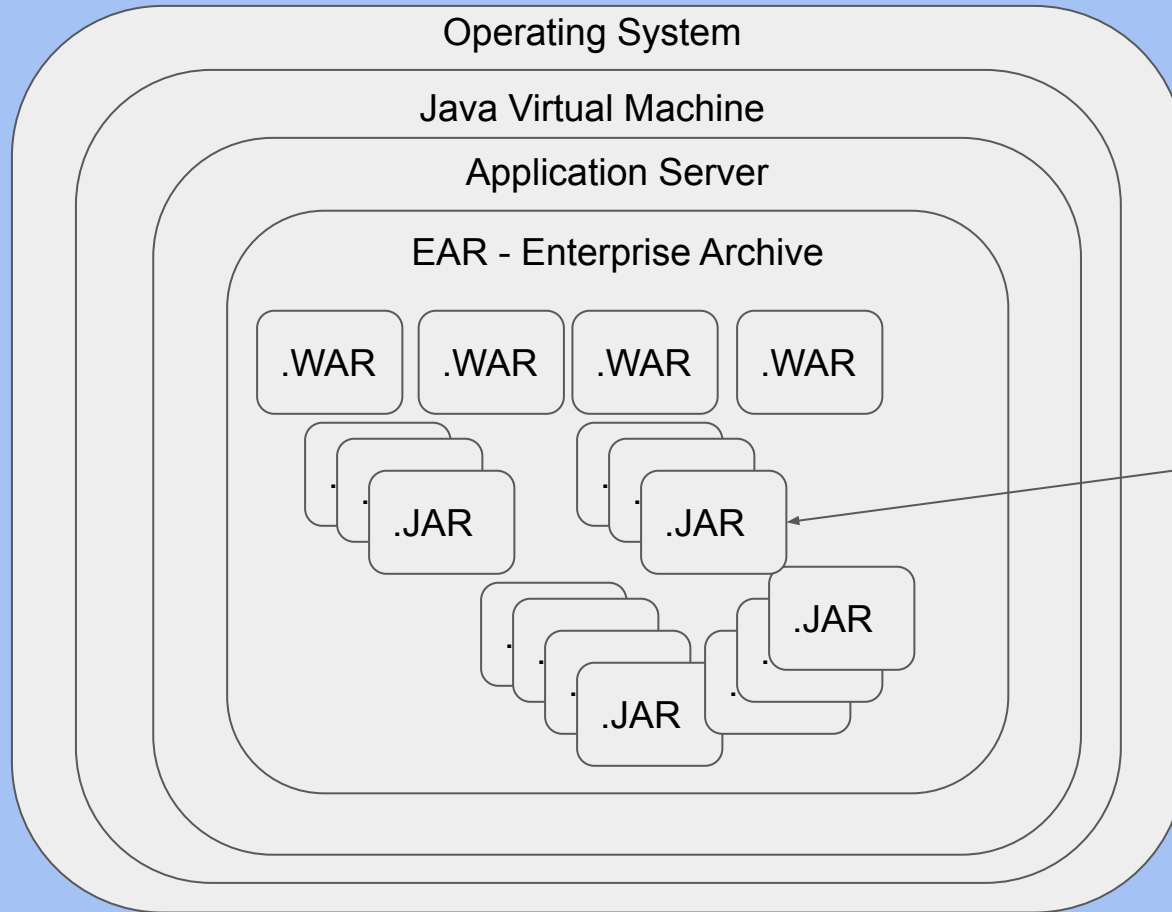
Week 6: B, F, I, J, G.1, A.3

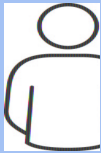
Week 7: A, C, D, E, G.2

Each microservice has its own independent team, practicing continuous delivery, typically deploying at the end of their independent 3-week sprints.

However, App and Stack patches may also need to be deployed mid-sprint.

Deployment frequency grows as organizational confidence grows





Programmers
(18)



Business
Analysts
(4)



Project
Managers
(2)



Quality
Assurance
(6)



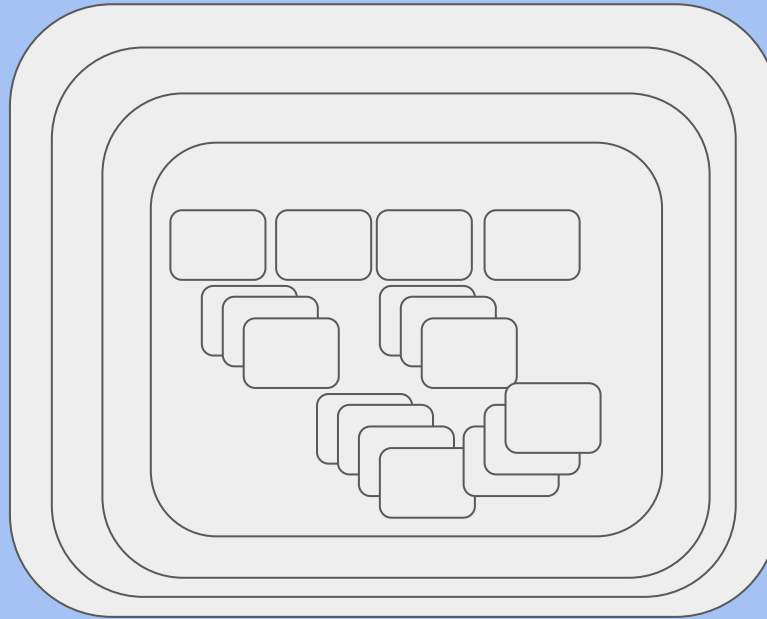
Security &
Compliance
(2)

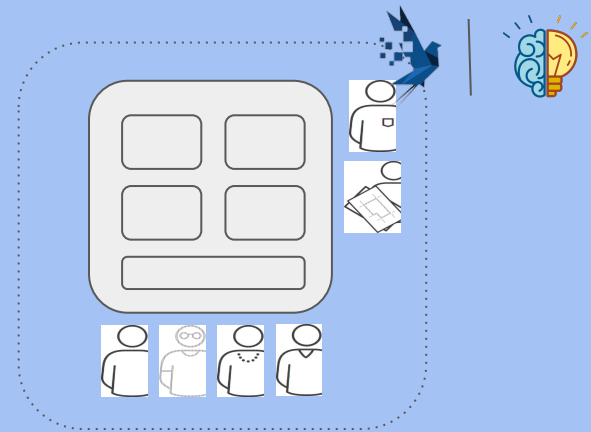
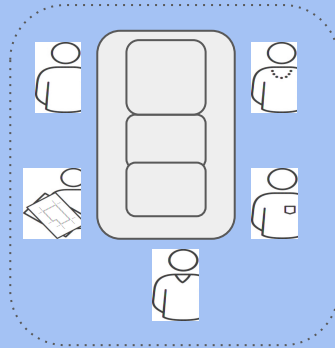
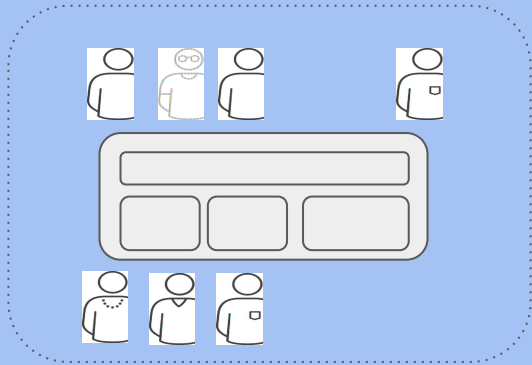


Operators
(6)

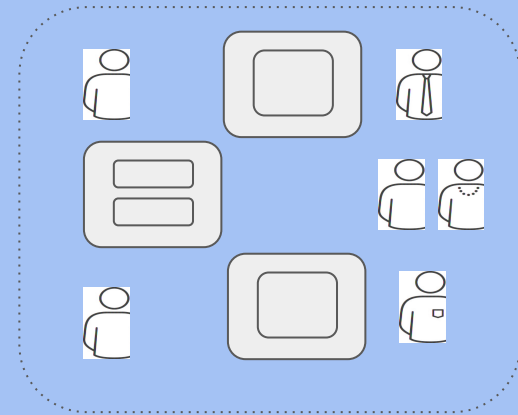
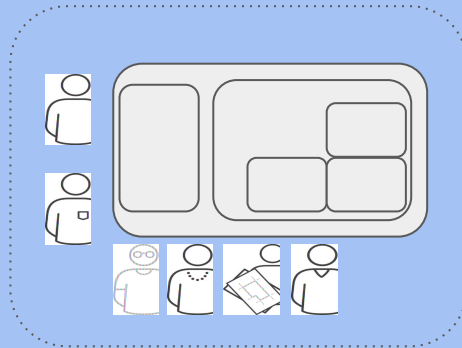
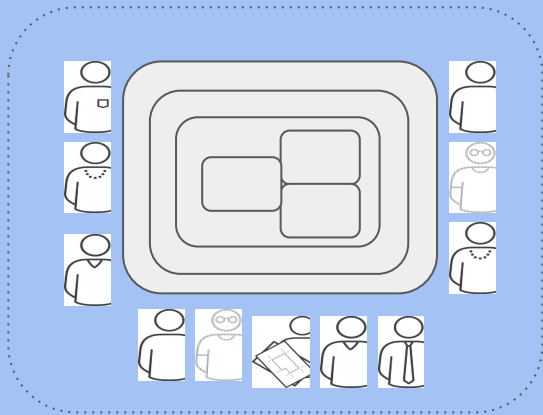


DBAs
(3)



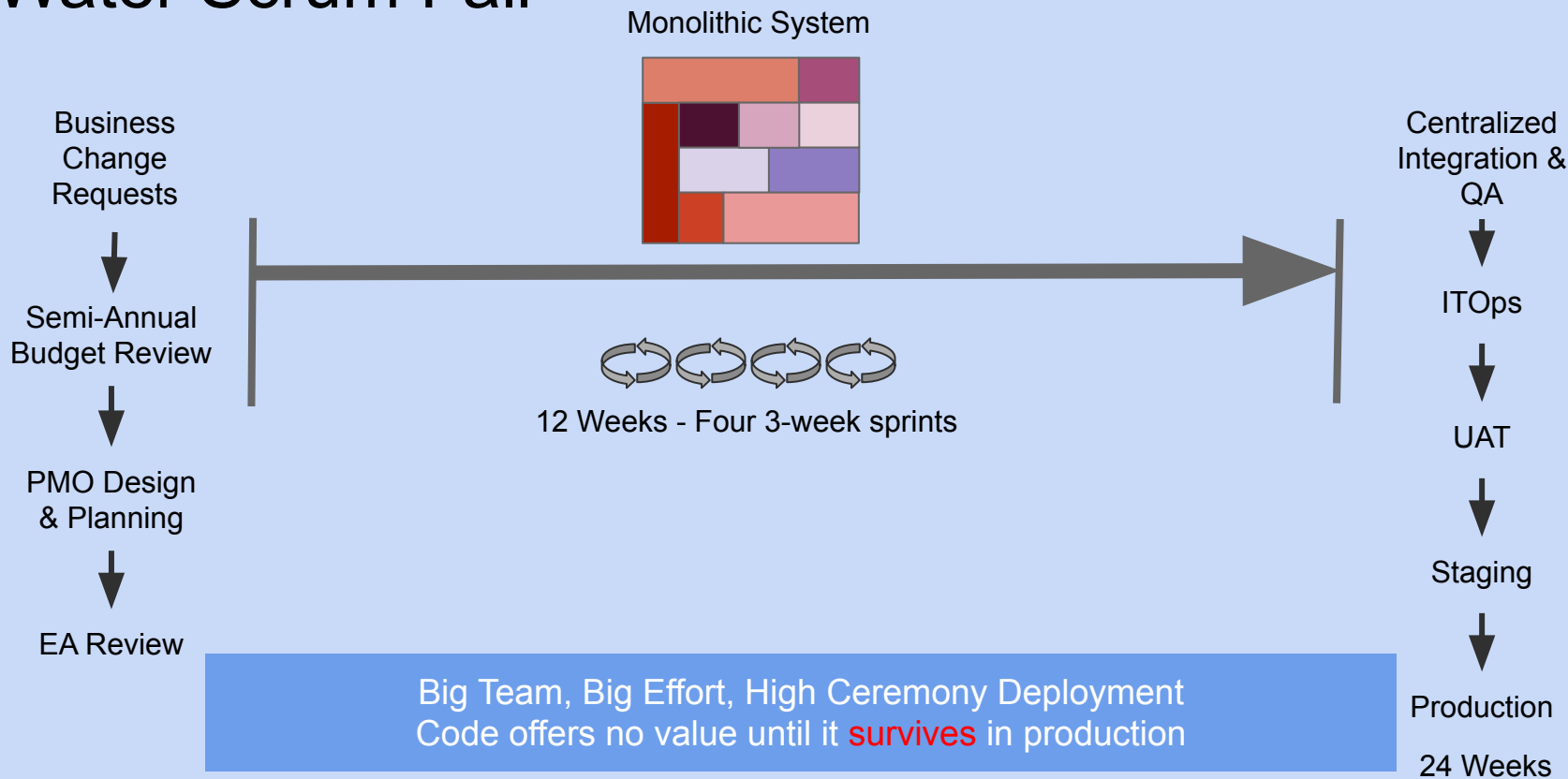


2-Pizza Teams You Build It, You Own it



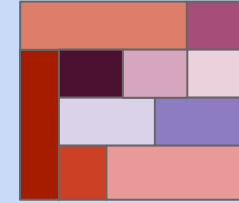


Water Scrum Fall





Monolithic System

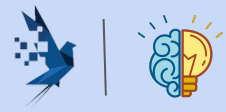


Business
Change
Requests

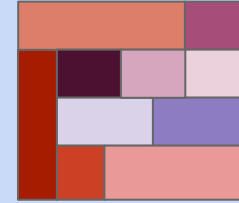


12 Weeks

Smaller Team, Smaller Effort, Lower Ceremony Deployment
Code offers no value until it **thrives** in production



Monolithic System

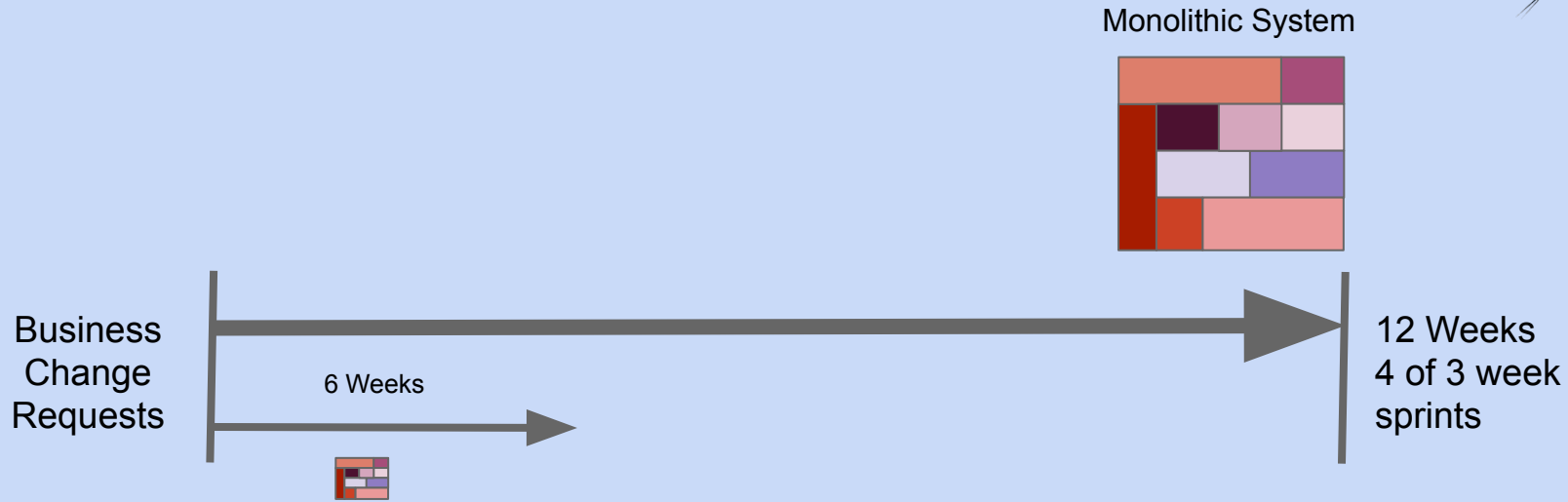


Business
Change
Requests

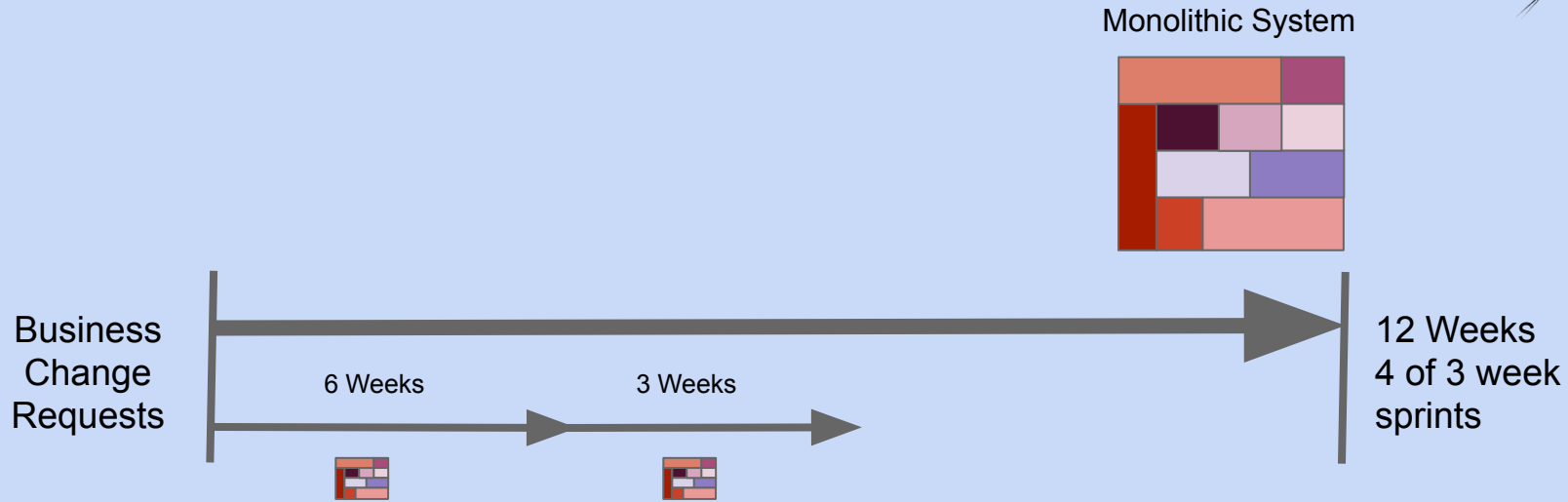


12 Weeks
4 of 3 week
sprints

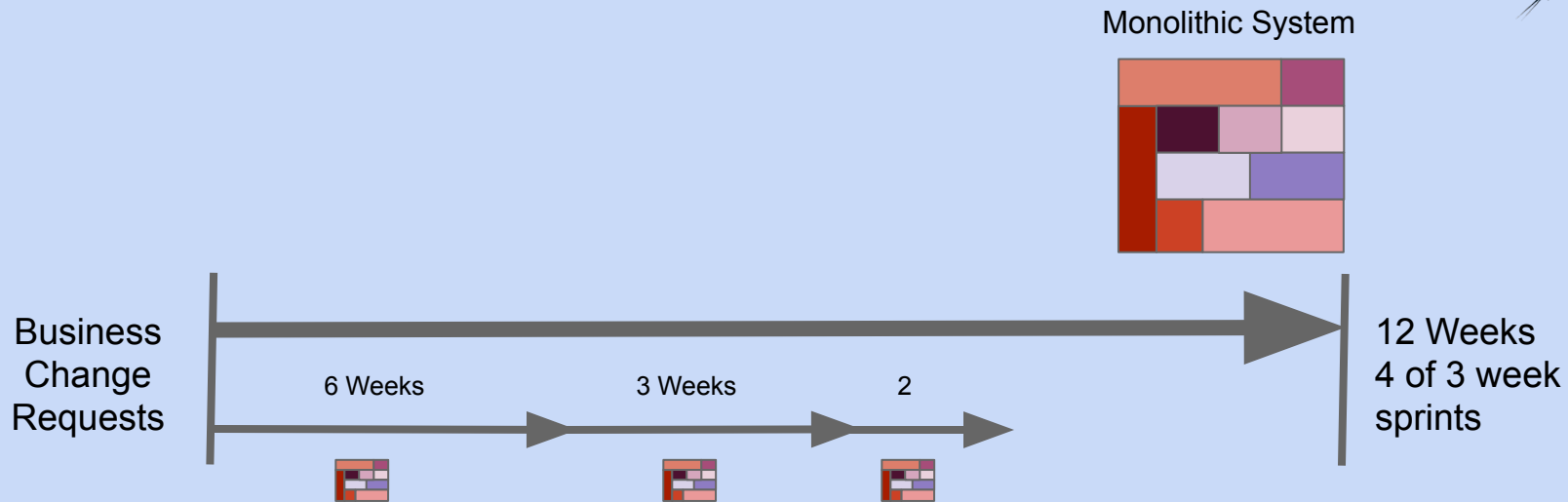
Getting a little Agile - thinking in small batches



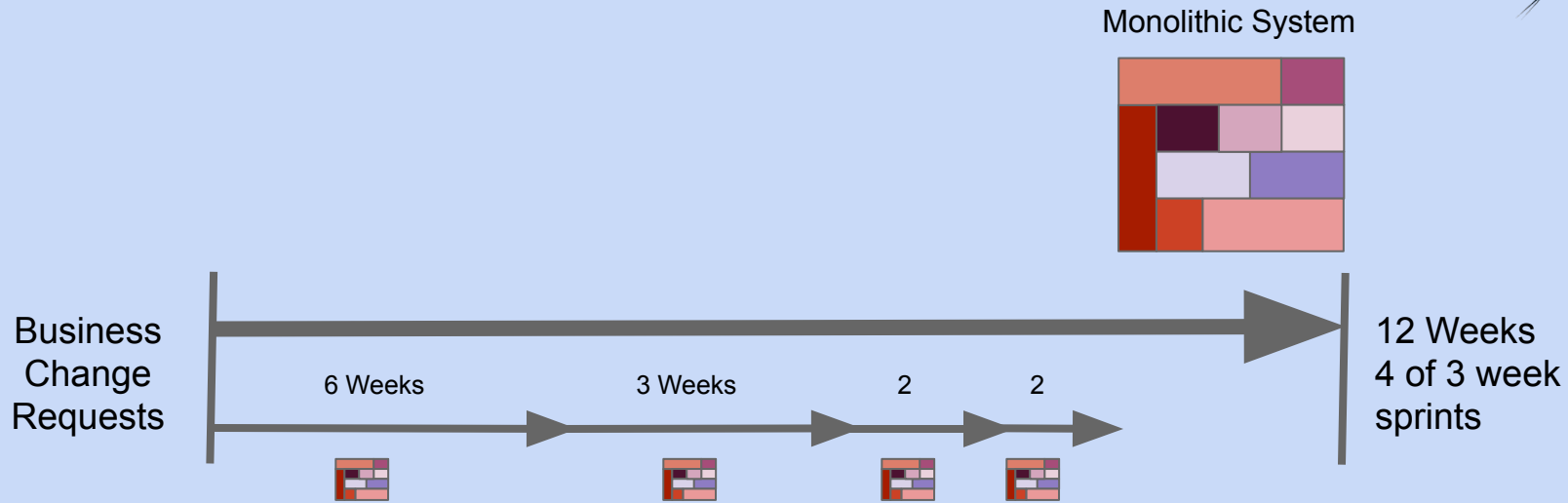
Automated Testing
Continuous Integration & Build Automation



Linux Containers (e.g. docker)
Automation via Orchestration (allows Devs to become DevOps)
Infrastructure as Code



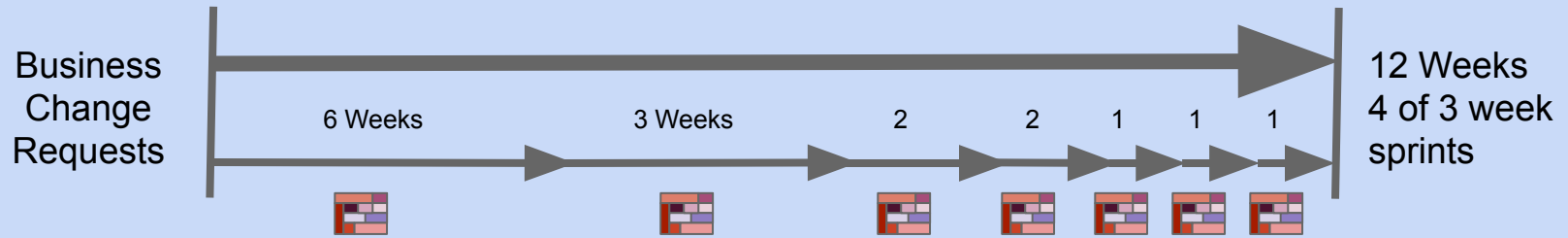
Continuous Delivery Pipeline



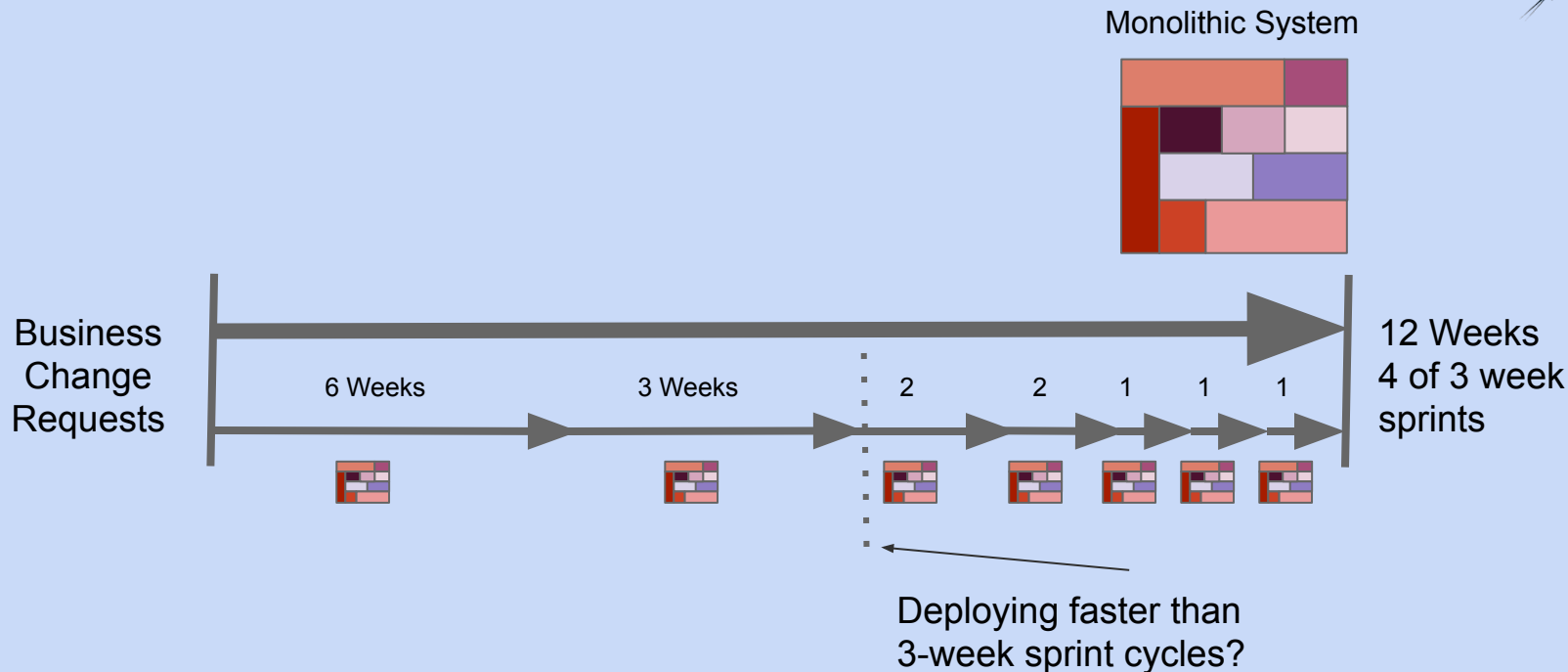
Zero-Downtime Deployment Strategies (Blue/Green, Canary)



Monolithic System



High Trust Environment



Patches to your application as well as your “stack” are also deployments. Your stack consisting of the OS, JVM, runtime engine (e.g. Tomcat, Vert.x, JBoss EAP), frameworks (e.g. Spring) all should be regularly patched via your CD Pipeline



Microservice Principles/Characteristics

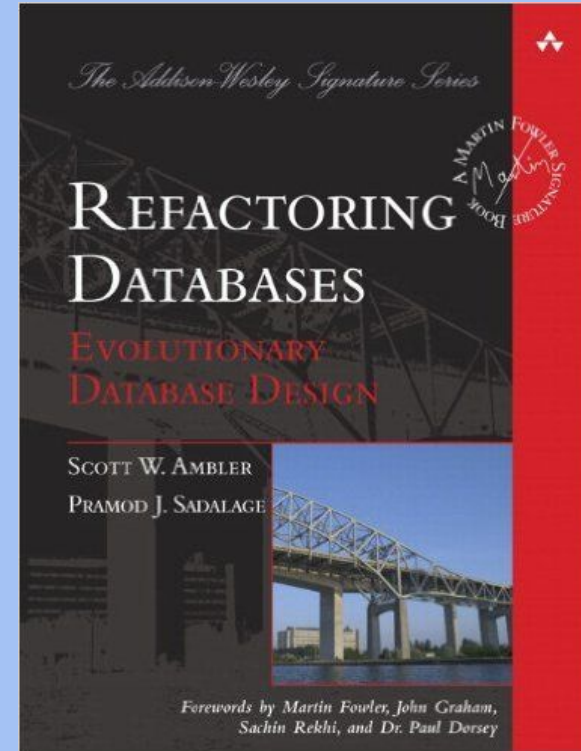
1. Deployment **Independence** - updates to an individual microservice have no negative impact to any other component of the system. Optimized for **Replacement**
2. Organized around **business** capabilities
3. **Products** not Projects
4. **API** Focused
5. **Smart** endpoints and dumb pipes
6. Decentralized Governance
7. Decentralized Data Management
8. Infrastructure Automation (infrastructure as code)
9. Design for failure
10. Evolutionary Design



Decentralized Data Management

Your Oracle DBA will hunt you down and hurt you...be ready.

Deployment independence requires owning your own schema



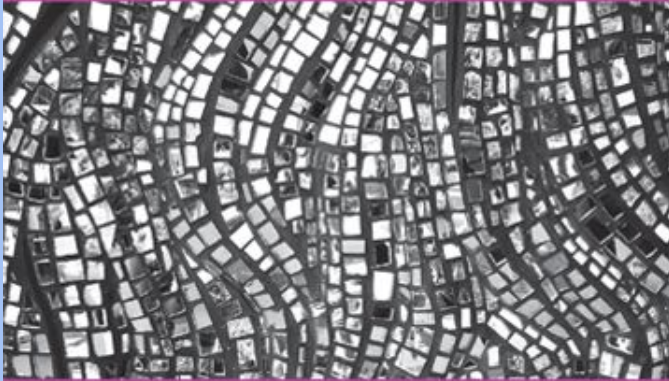
O'REILLY®



Compliments of
RED HAT
DEVELOPERS

Migrating to Microservice Databases

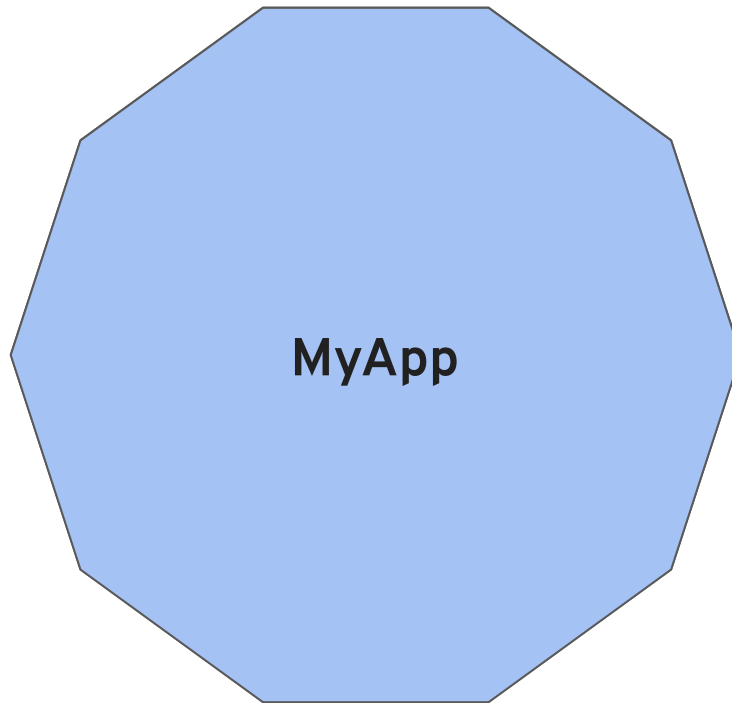
From Relational Monolith
to Distributed Data



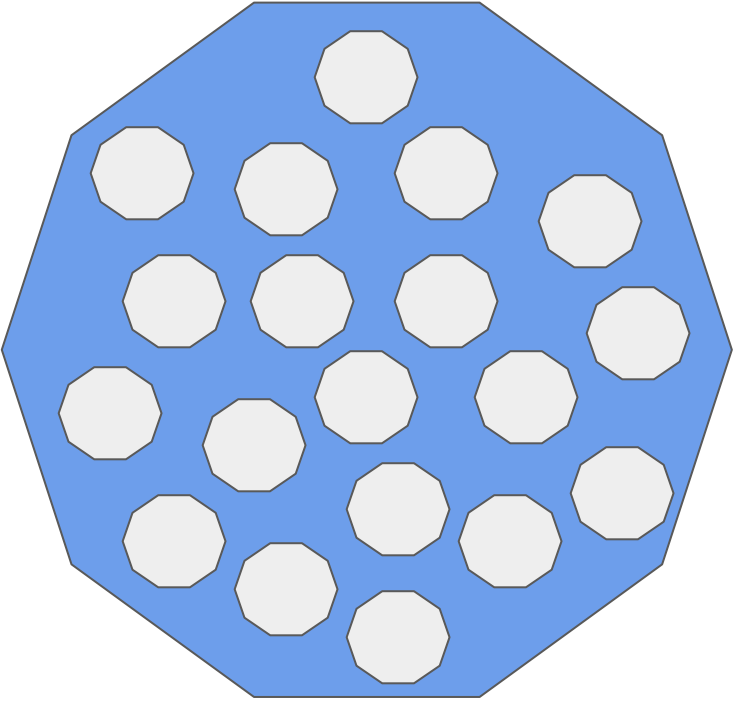
Edson Yanaga



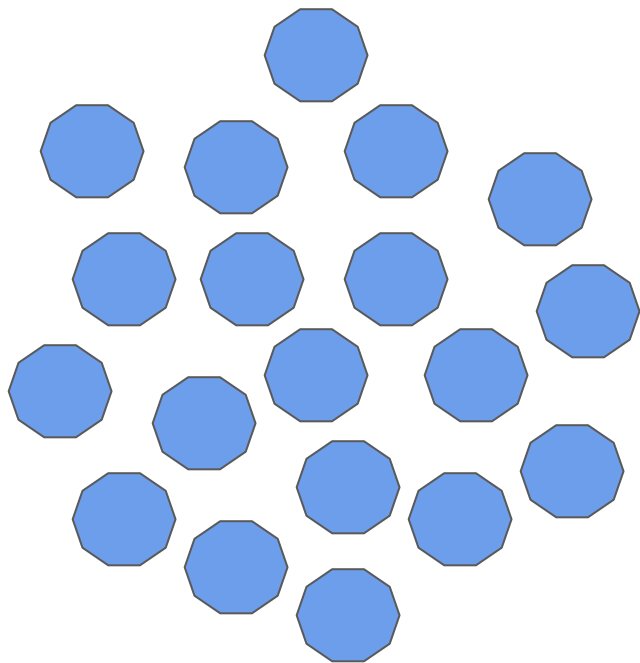
Monolith



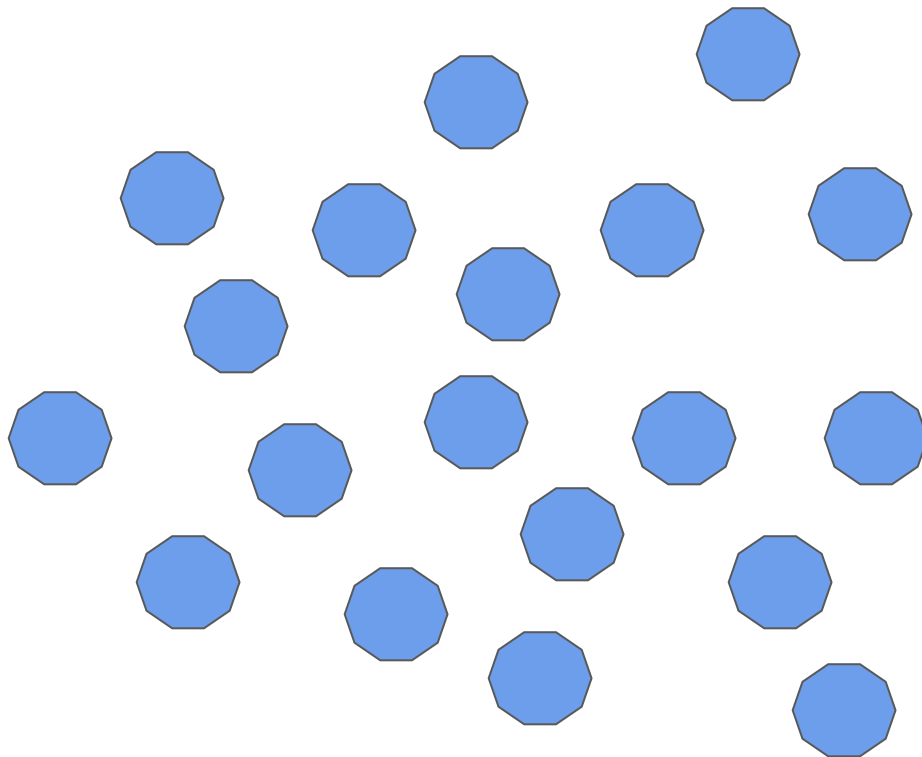
Microservices



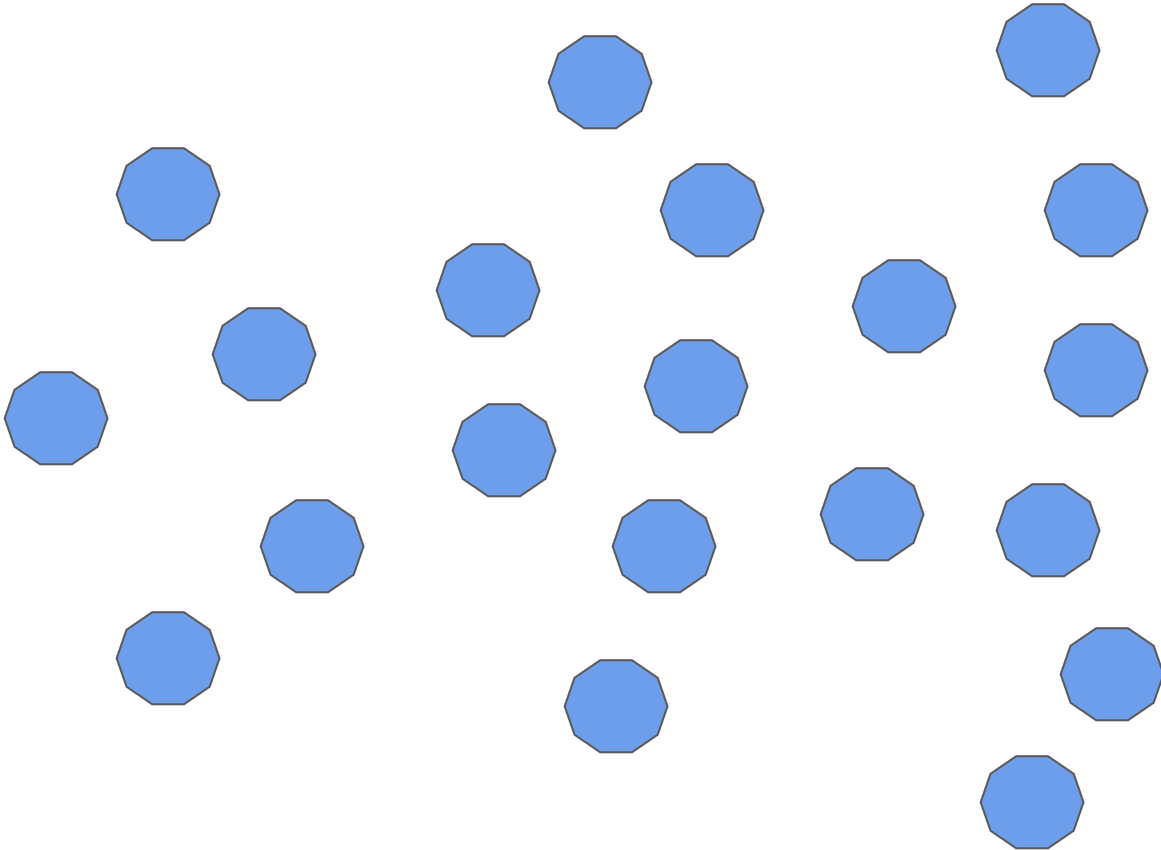
Microservices



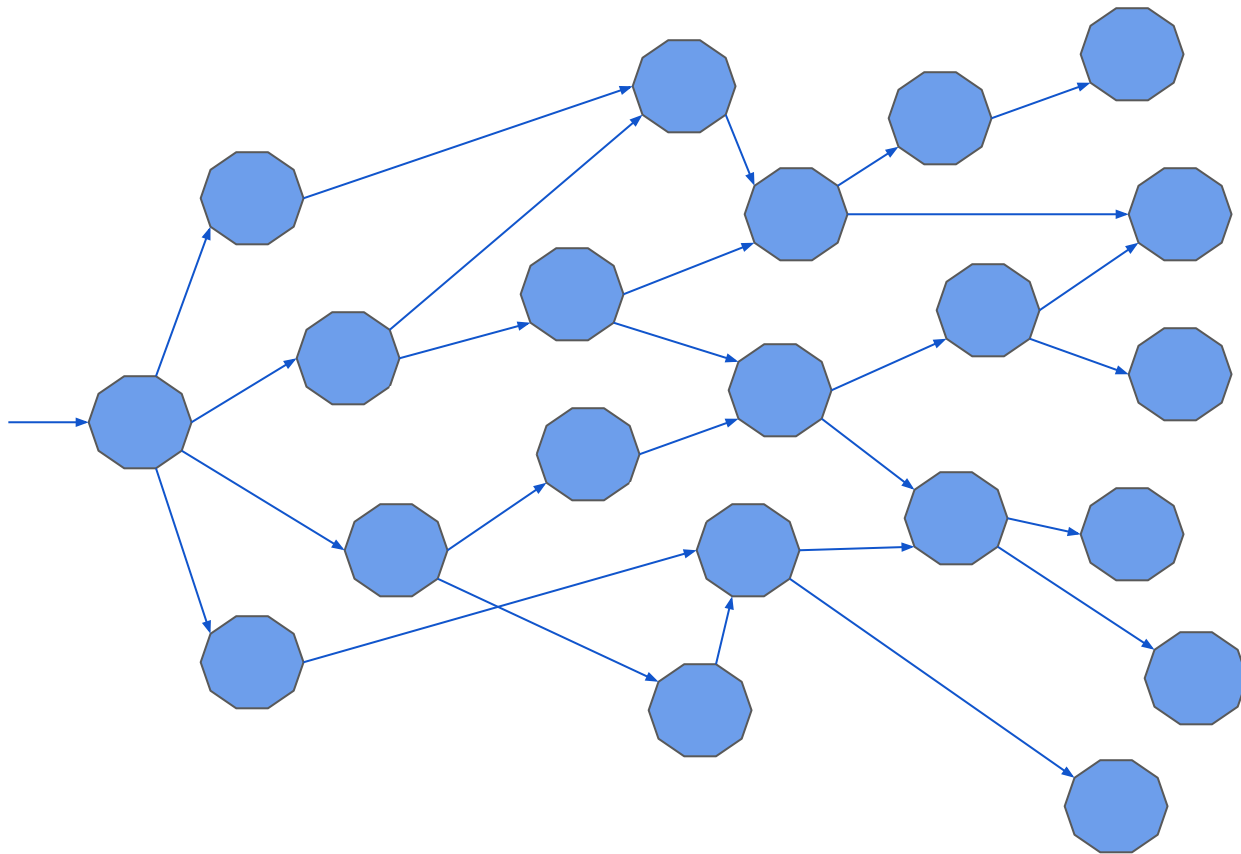
Microservices



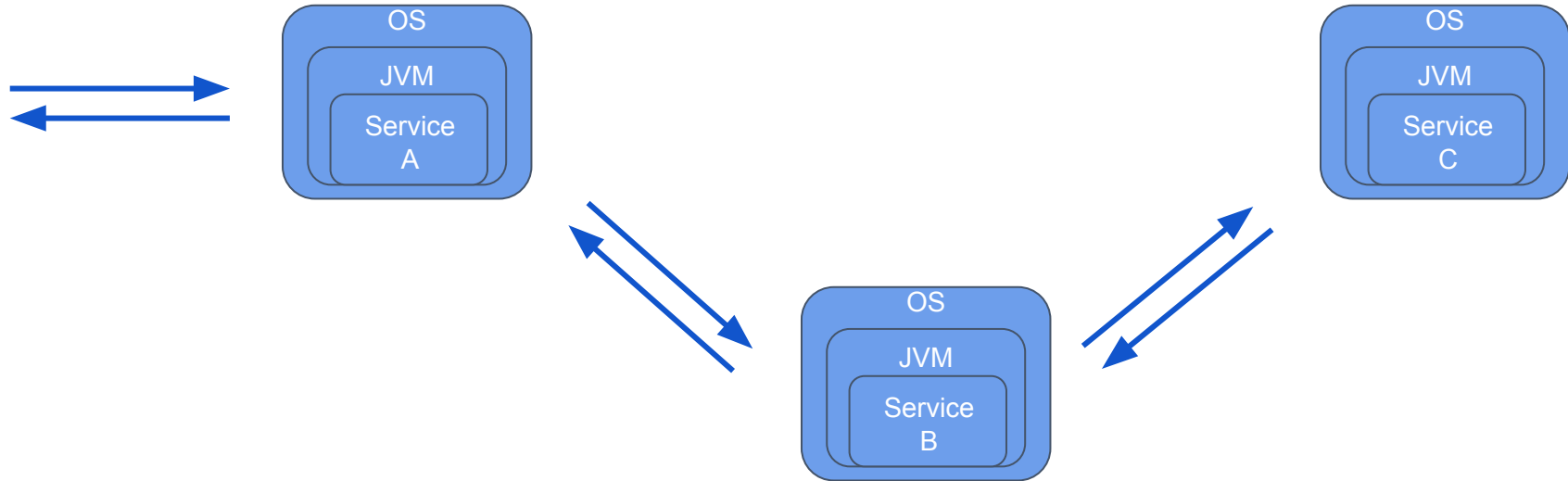
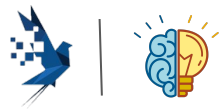
Microservices



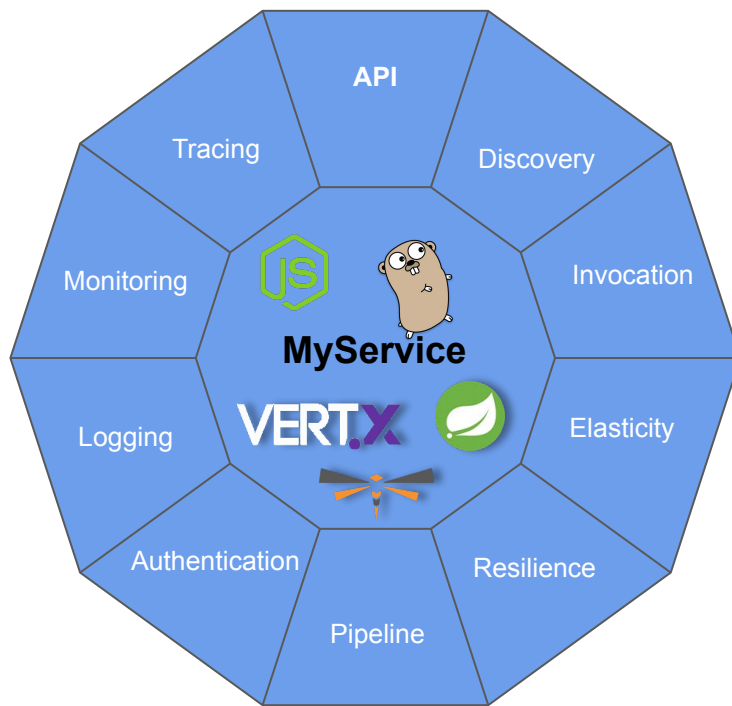
Network of Services



Microservices == Distributed Computing



Microservices'ilities





Short History of Microservices



NETFLIX

OSS



What's Wrong with Netflix OSS?

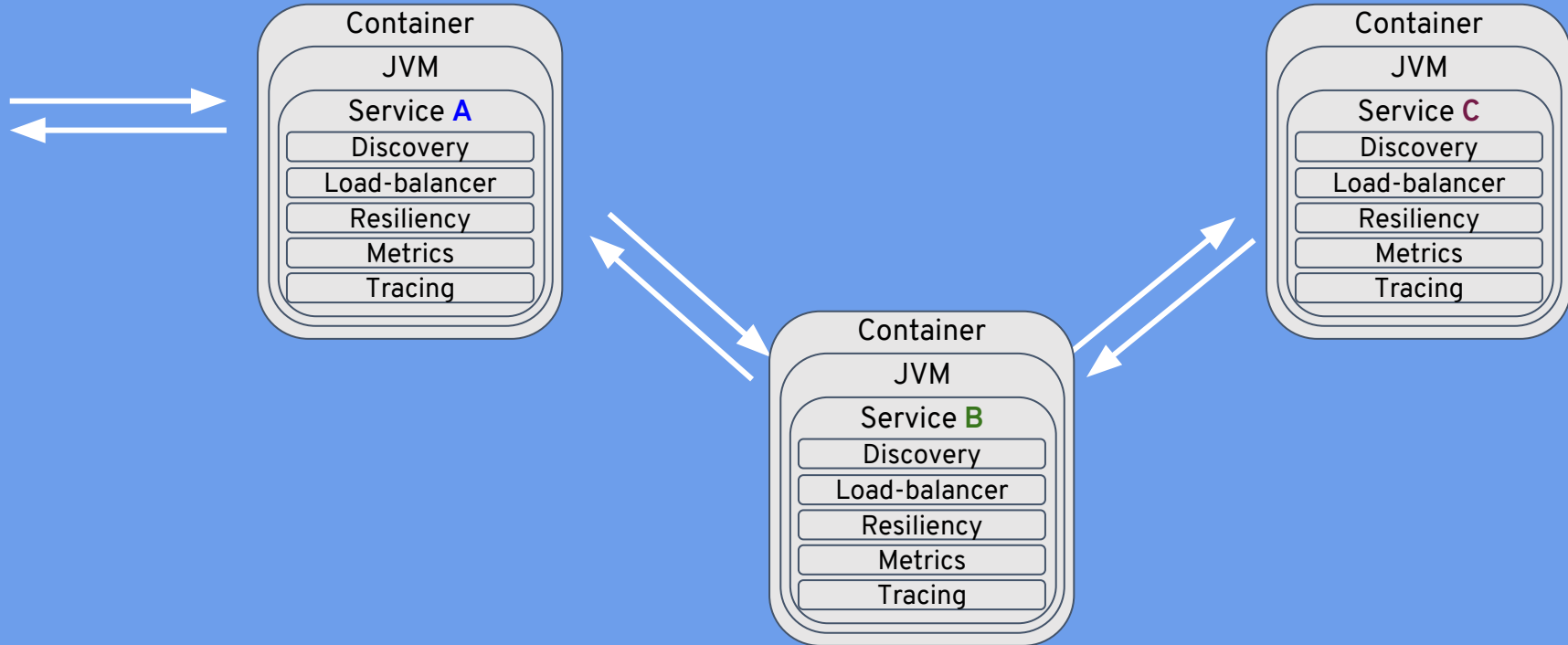
Java Only

Adds a lot of libraries to YOUR code

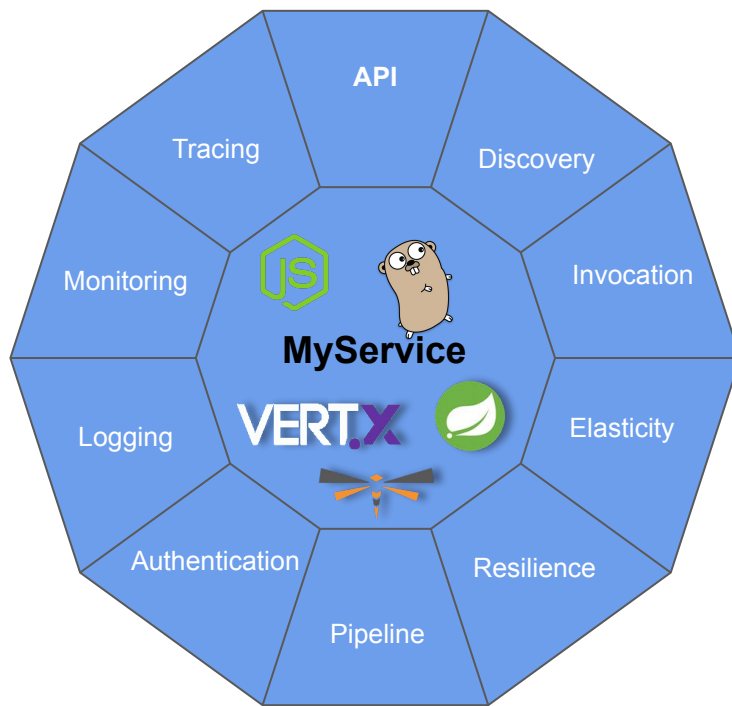


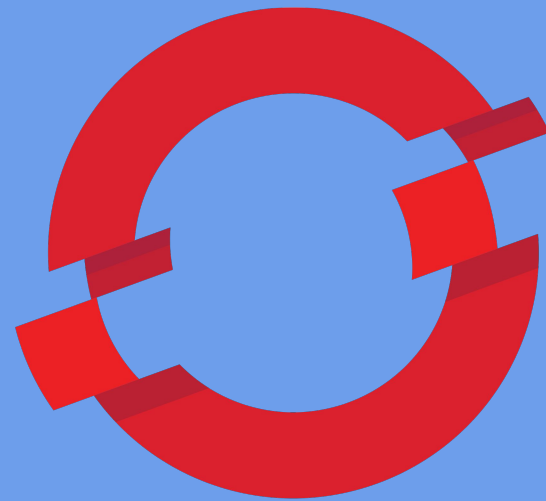


Microservices embedding Capabilities



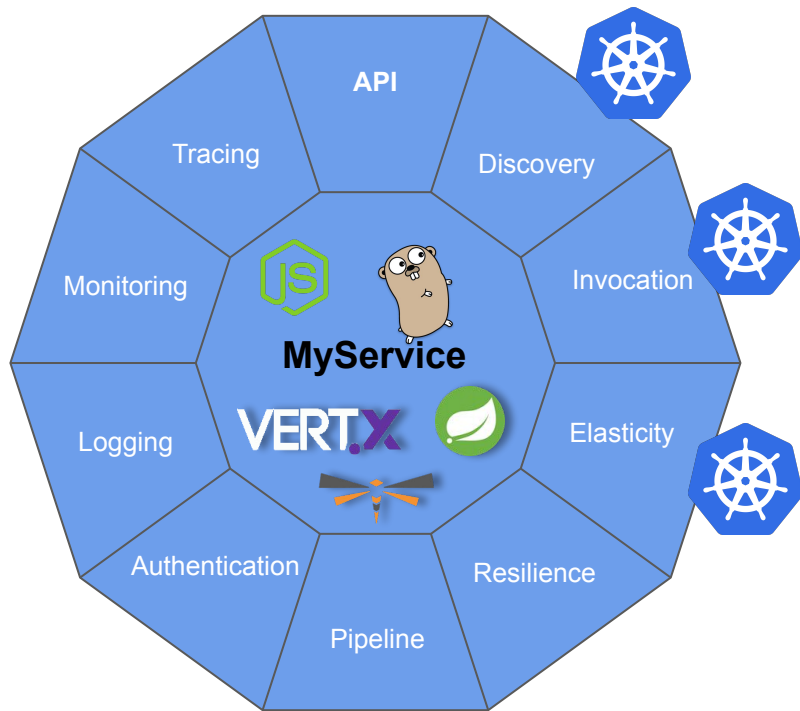
Microservices'ilities



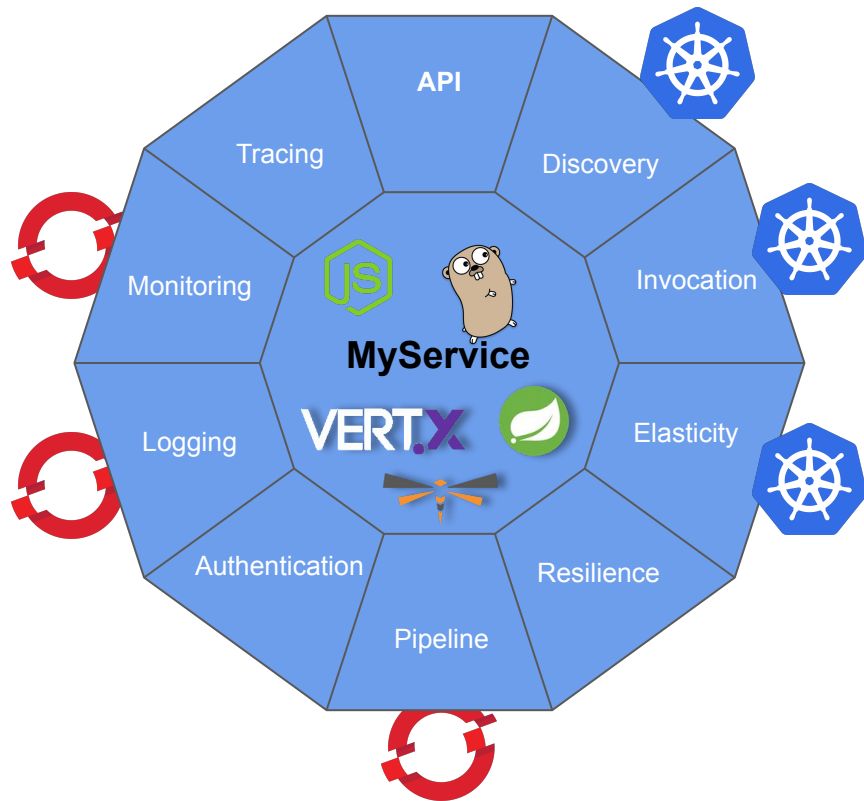


OPENSIFT

Microservices'ilities + Kubernetes



Microservices'ilities + OpenShift





Istio - Sail

(Kubernetes - Helmsman or ship's pilot)

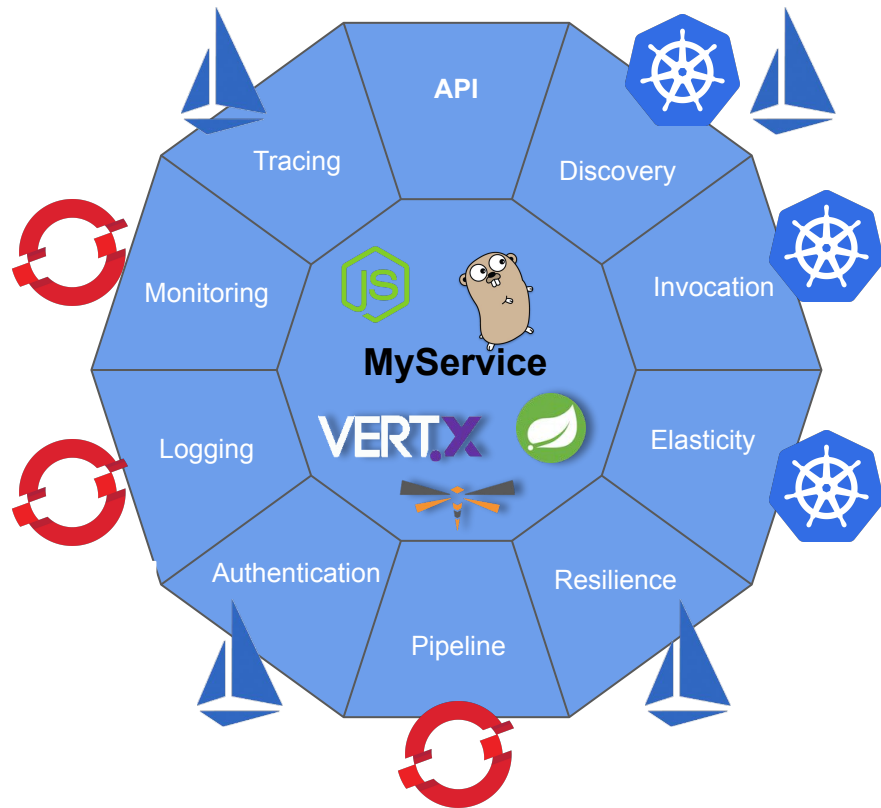


Service Mesh Defined

A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware

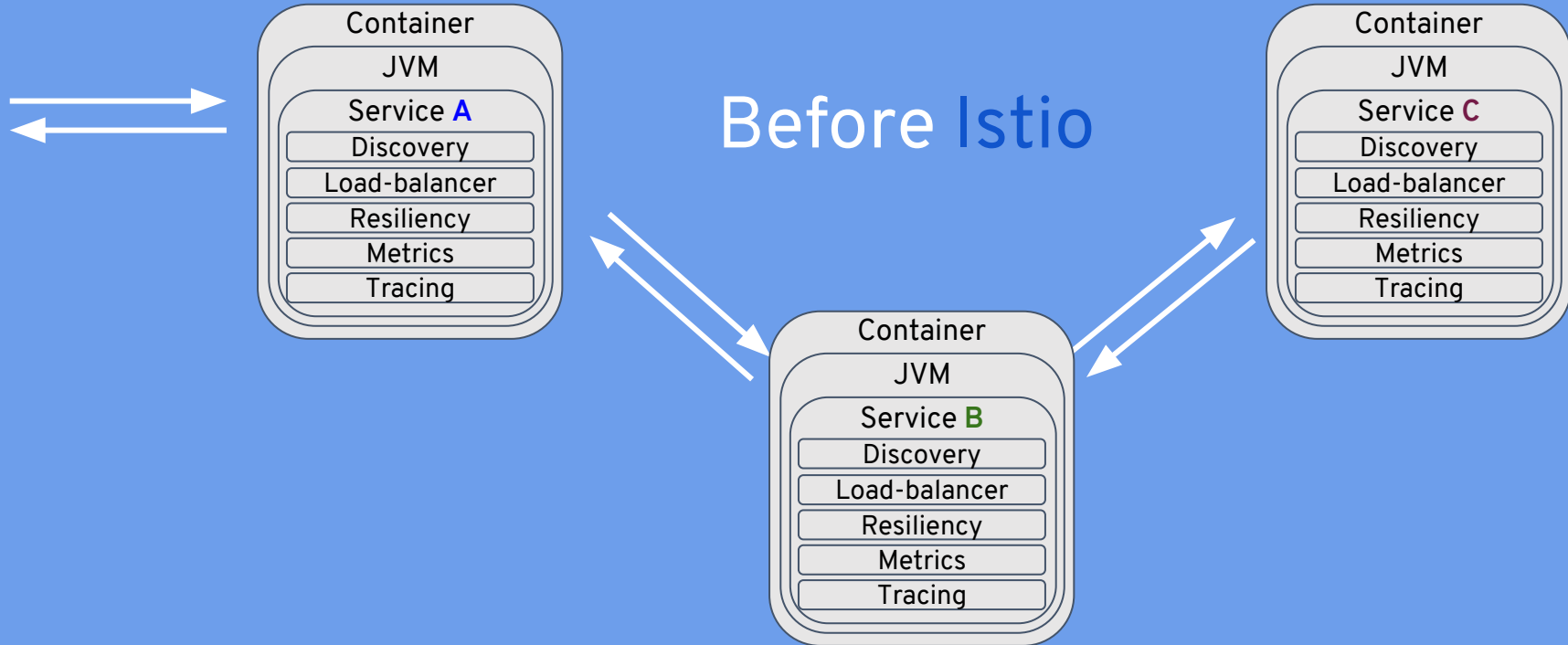
<https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/>

Microservices'ilities + Istio



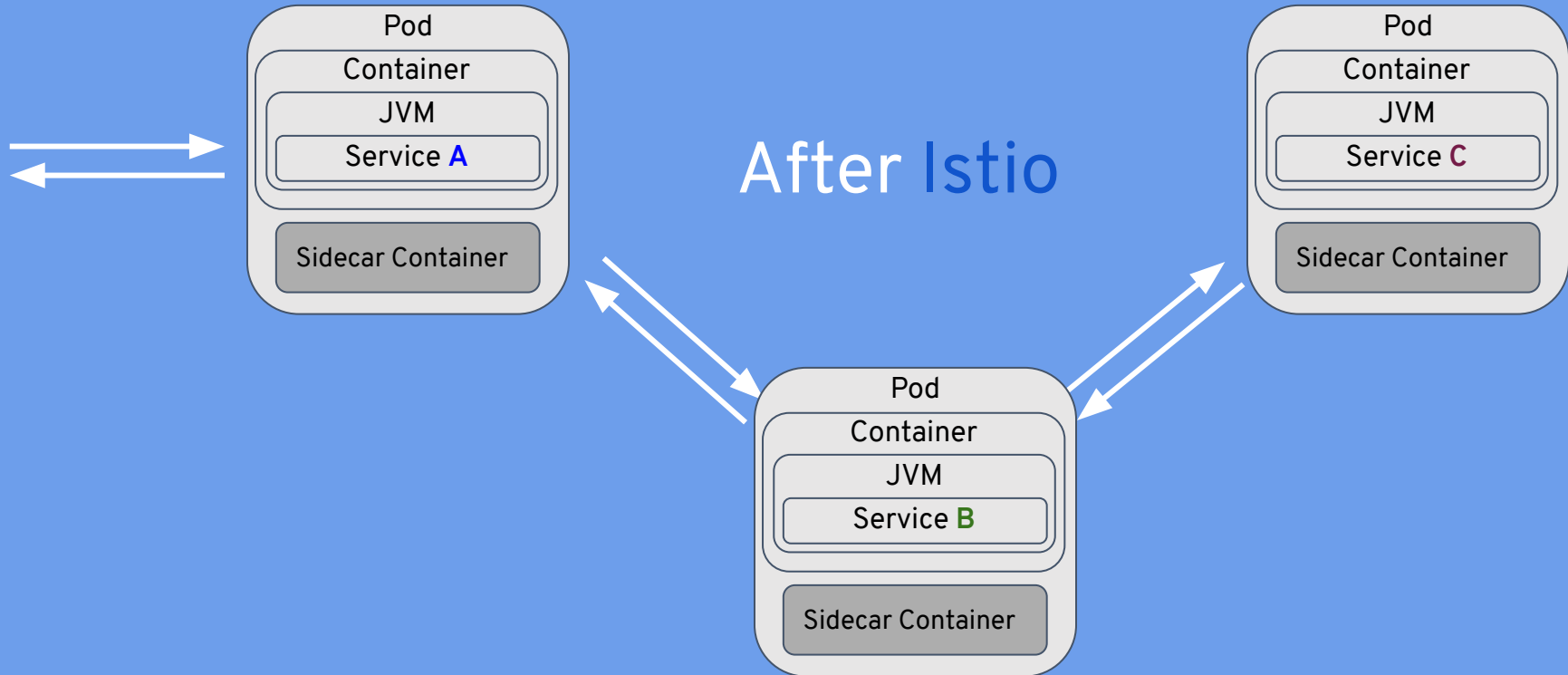


Microservices embedding Capabilities

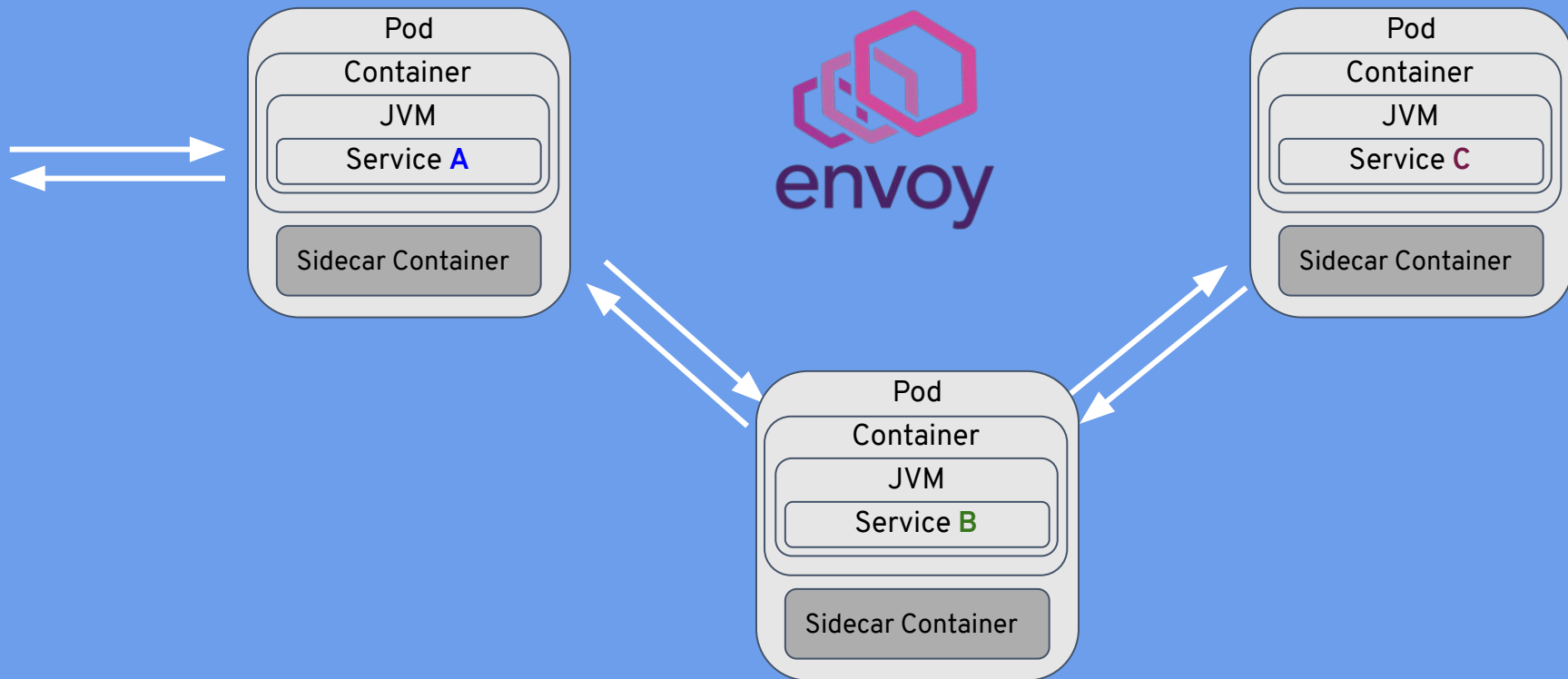




Microservices externalizing Capabilities



Kubernetes, Istio, Envoy



Next Generation Microservices - Service Mesh



Code Independent (Polyglot)

- Intelligent Routing and Load-Balancing
 - A/B Tests
 - Smarter Canary Releases
- Chaos: Fault Injection
- Resilience: Circuit Breakers
- Observability: Metrics and Tracing
- Fleet wide policy enforcement



bookinfo_istio.yaml ↔ bookinfo.yaml — bookinfo

EXPLORER

- ! bookinfo.yaml
- ! bookinfo_istio.yaml
- ! bookinfo_istio.yaml ↔ bookinfo...
- BOOKINFO
 - ! bookinfo_istio.yaml
 - ! bookinfo-ingress.yaml
 - ! bookinfo-v1.yaml
 - ! bookinfo.yaml
 - cleanup.sh
 - ! destination-ratings-test-delay.yaml
 - ! loadbalancing-policy-reviews.yaml
 - ! mixer-rule-additional-telemetry.ya...
 - ! mixer-rule-empty-rule.yaml
 - ! mixer-rule-ratings-denial.yaml
 - ! mixer-rule-ratings-ratelimit.yaml
 - README.md
 - ! route-rule-all-v1.yaml
 - ! route-rule-delay.yaml
 - ! route-rule-reviews-50-v3.yaml
 - ! route-rule-reviews-test-v2.yaml
 - ! route-rule-reviews-v2-v3.yaml
 - ! route-rule-reviews-v3.yaml

! bookinfo_istio.yaml

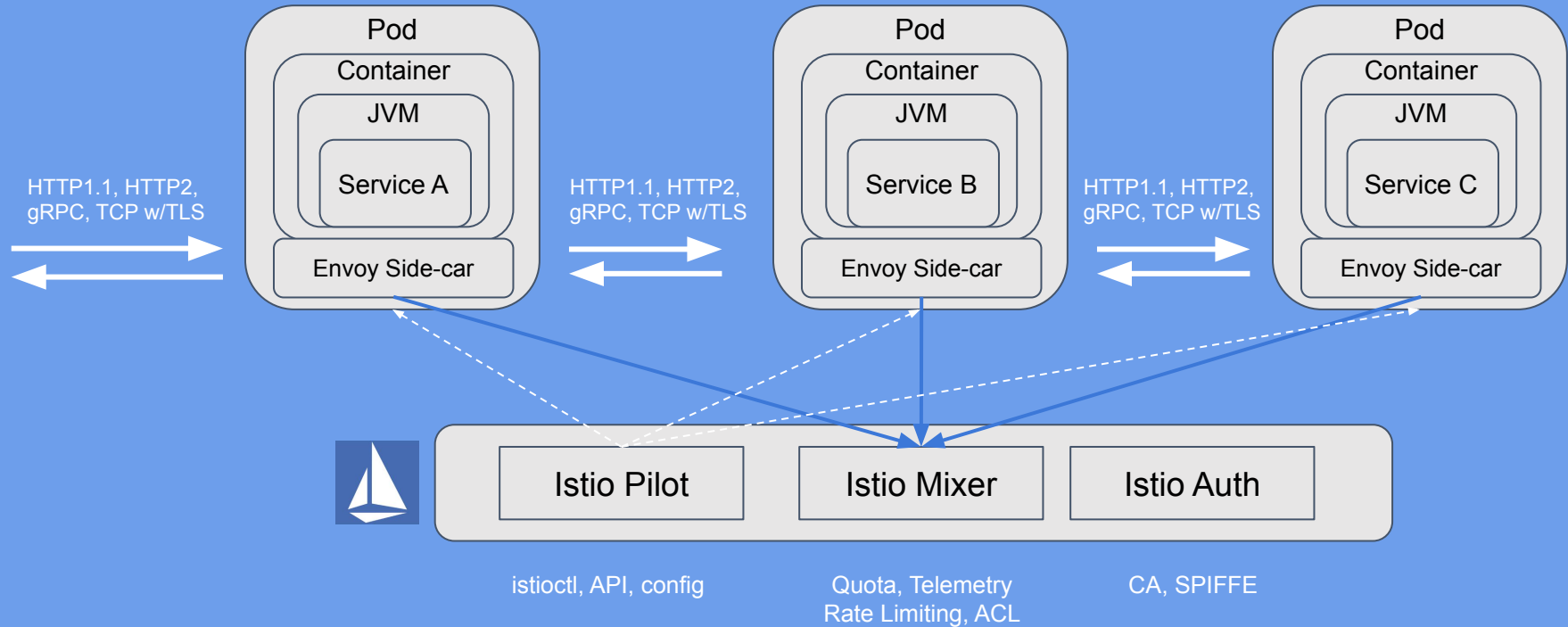
```
41 - annotations:
42 -   alpha.istio.io/sidecar: injected
43 -   alpha.istio.io/version: jenkins@ubuntu-16-04-build
44 -   pod.beta.kubernetes.io/init-containers: '[{"args":
45 -     -w kernel.core_pattern=/tmp/core.%e.%p.%t \u0026
46 -   creationTimestamp: null
47 -   labels:
48 -     app: details
49 -     version: v1
50 -   spec:
51 -     containers:
52 -       - image: istio/examples-bookinfo-details-v1
53 -         imagePullPolicy: IfNotPresent
54 -         name: details
55 -         ports:
56 -           - containerPort: 9080
57 -         resources: {}
58 -       - args:
59 -         - proxy
60 -         - sidecar
61 -         - -v
62 -         - "2"
63 -       env:
64 -         - name: POD_NAME
65 -           valueFrom:
66 -             fieldRef:
67 -               fieldPath: metadata.name
68 -         - name: POD_NAMESPACE
69 -           valueFrom:
70 -             fieldRef:
71 -               fieldPath: metadata.namespace
72 -         - name: POD_IP
73 -           valueFrom:
74 -             fieldRef:
75 -               fieldPath: status.podIP
76 -         image: docker.io/istio/proxy_debug:0.1
77 -         imagePullPolicy: Always
```

! bookinfo_istio.yaml ↔ bookinfo.yaml x

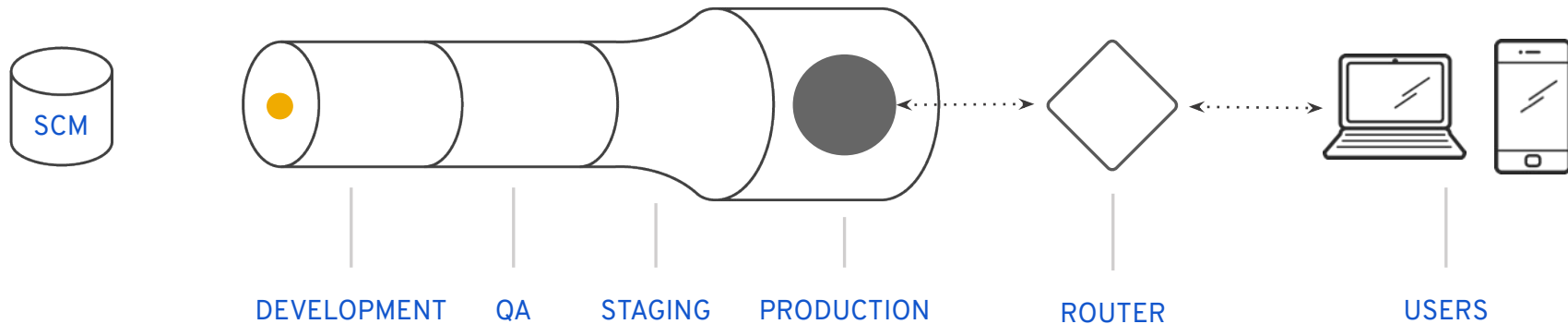
```
39 labels:
40   app: details
41   version: v1
42 spec:
43   containers:
44 +   - name: details
45 +     image: istio/examples-bookinfo-details-v1
46     imagePullPolicy: IfNotPresent
47   ports:
48   - containerPort: 9080
```



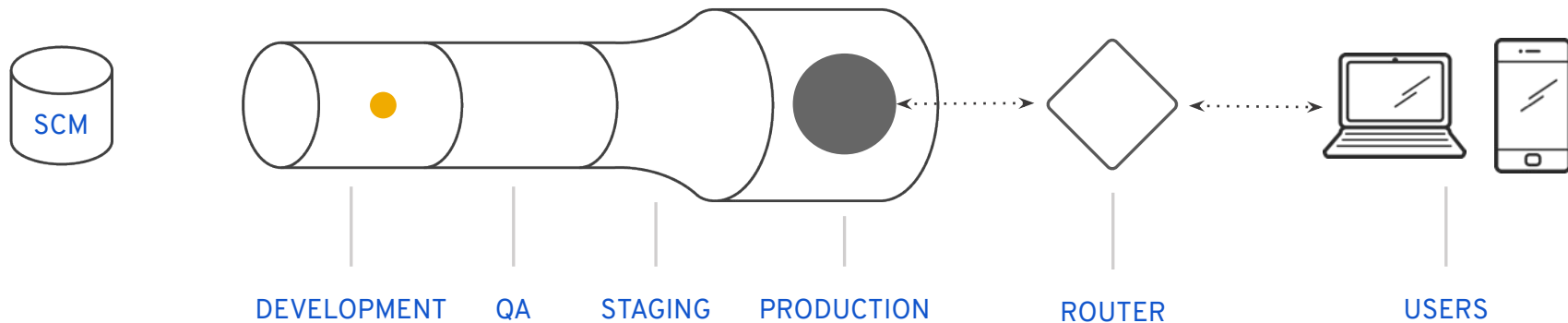
Istio Control Plane



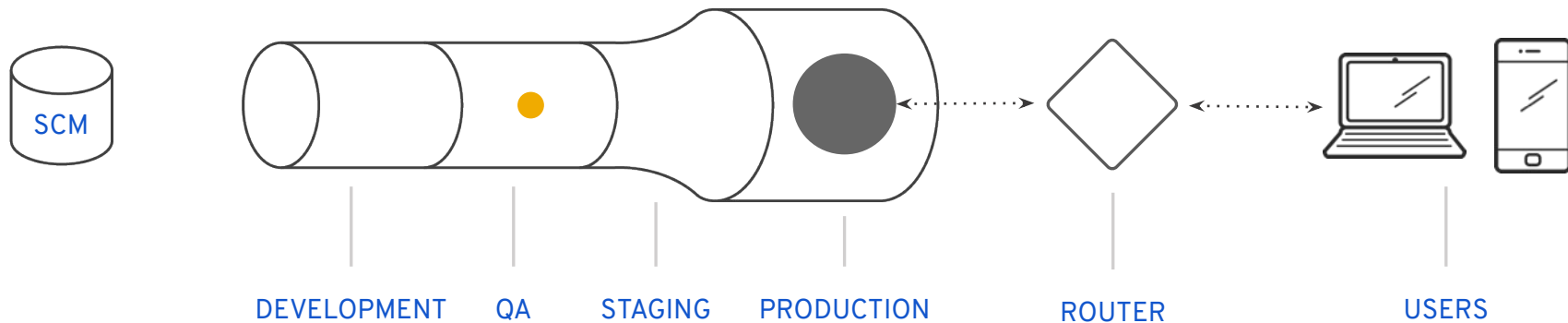
Canary Deployment



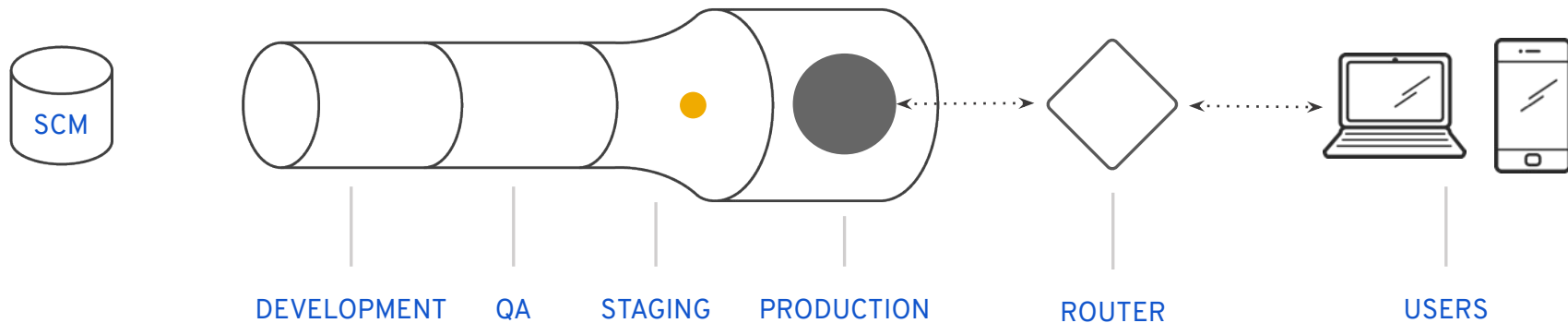
Canary Deployment



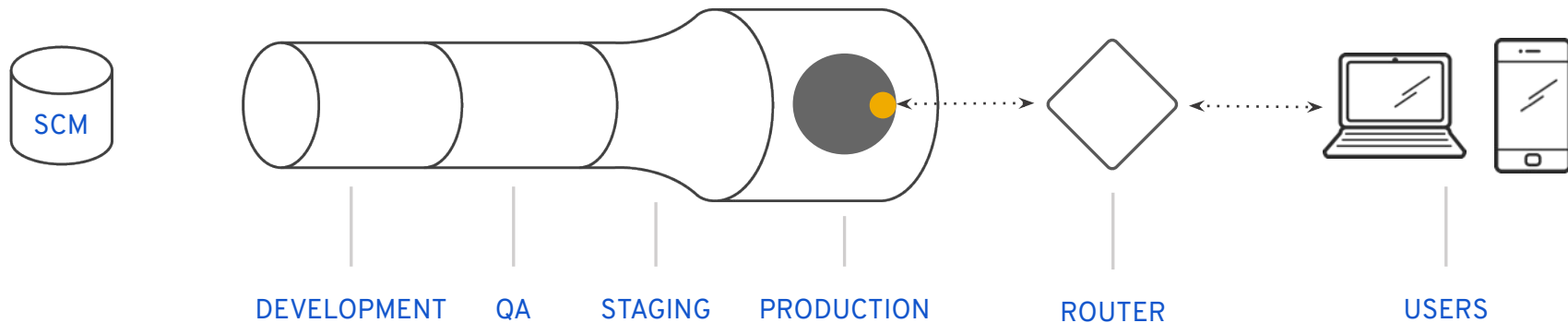
Canary Deployment



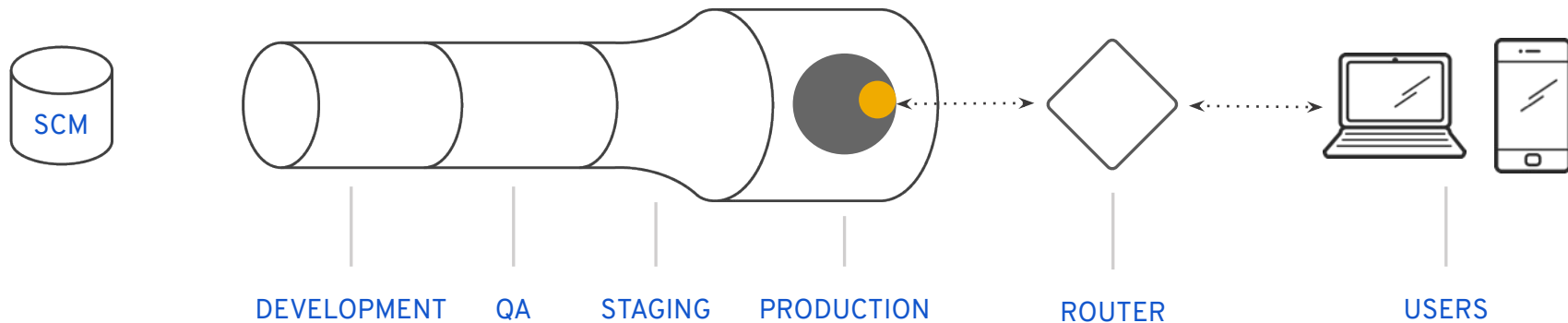
Canary Deployment



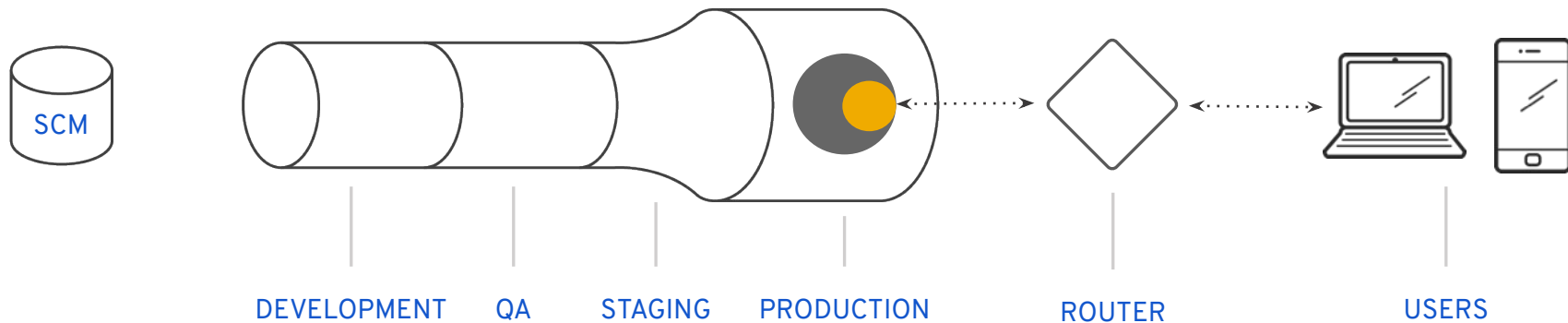
Canary Deployment



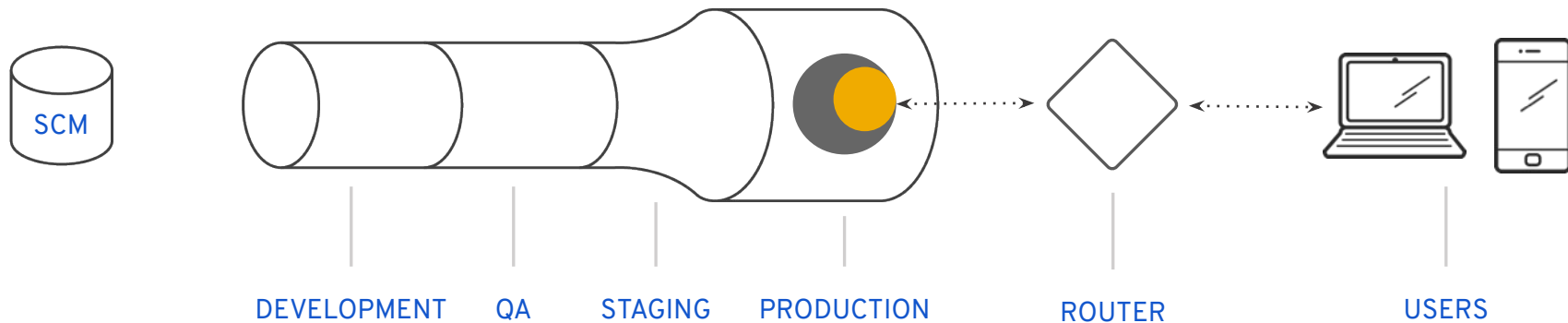
Canary Deployment



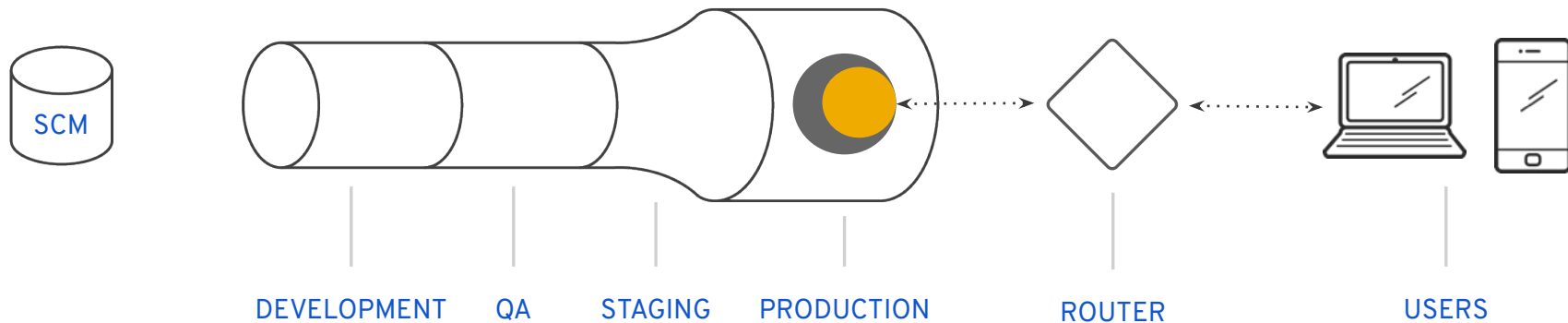
Canary Deployment



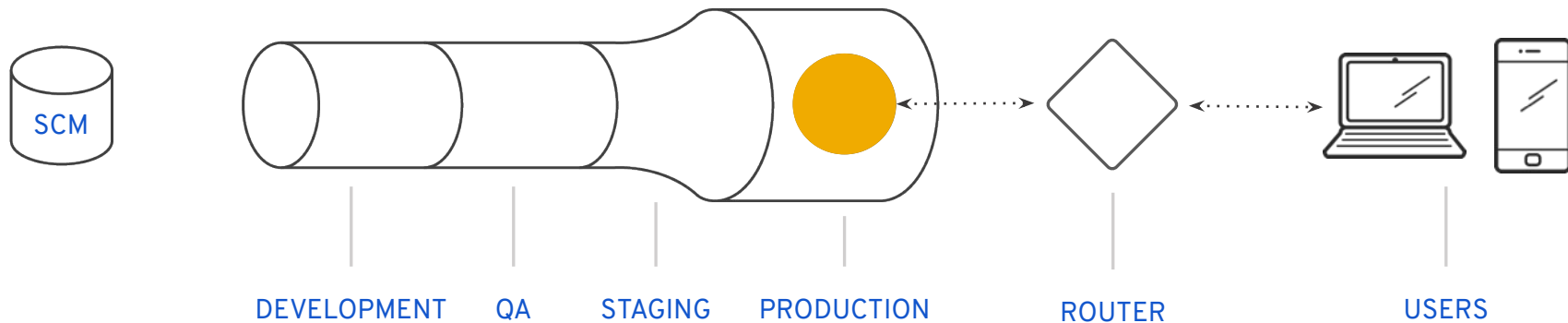
Canary Deployment



Canary Deployment

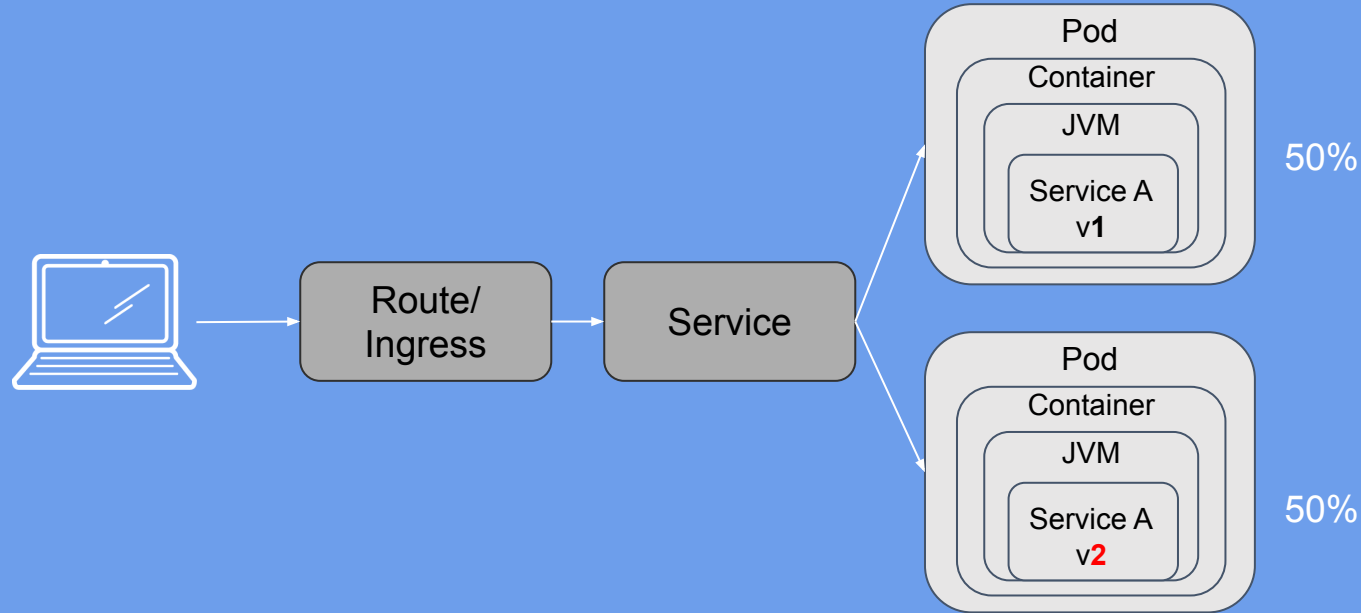


Canary Deployment

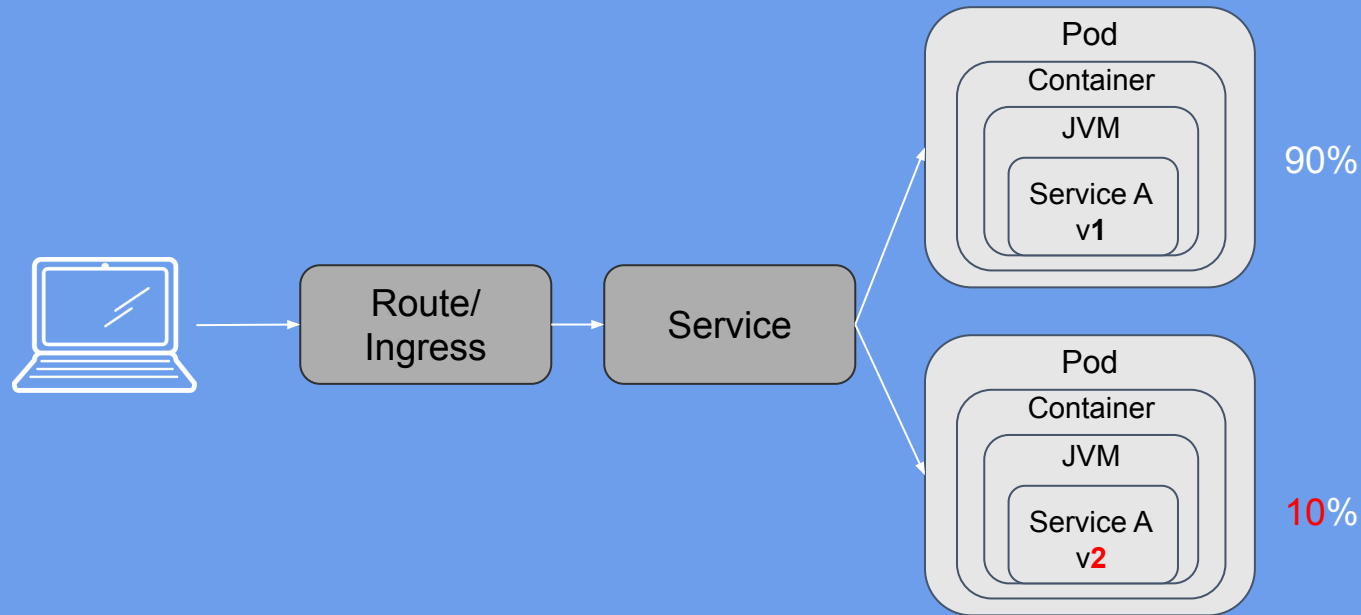




Canaries with Kubernetes



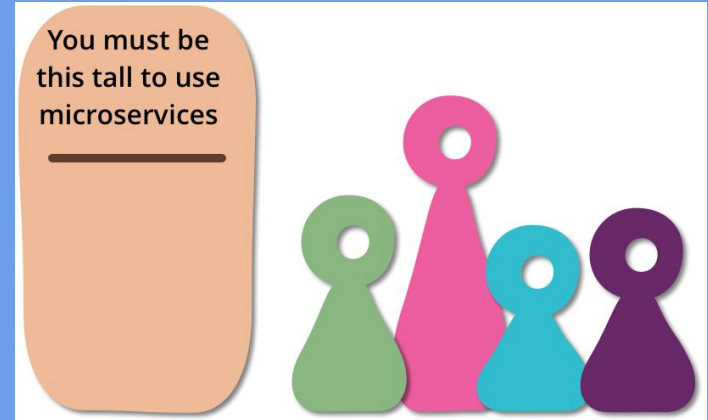
Canaries with Istio





You Must Be This Tall

1. Self-Service, on-demand, elastic infrastructure as code
(how many days/weeks to provision a new VM?)
2. Dev vs Ops
(who is on the pager for production app outage?)
3. Automation
(phoenix vs snowflake?)
4. CI & CD
5. Deployment Pipeline

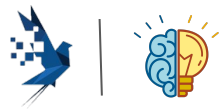


Any Queries?



**ANY
QUESTIONS?**

Thank You...



**THANK
YOU!**

