

Path to Resilient and Observable Microservices

Slides: <https://slides.peterj.dev>

Safe Harbor

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

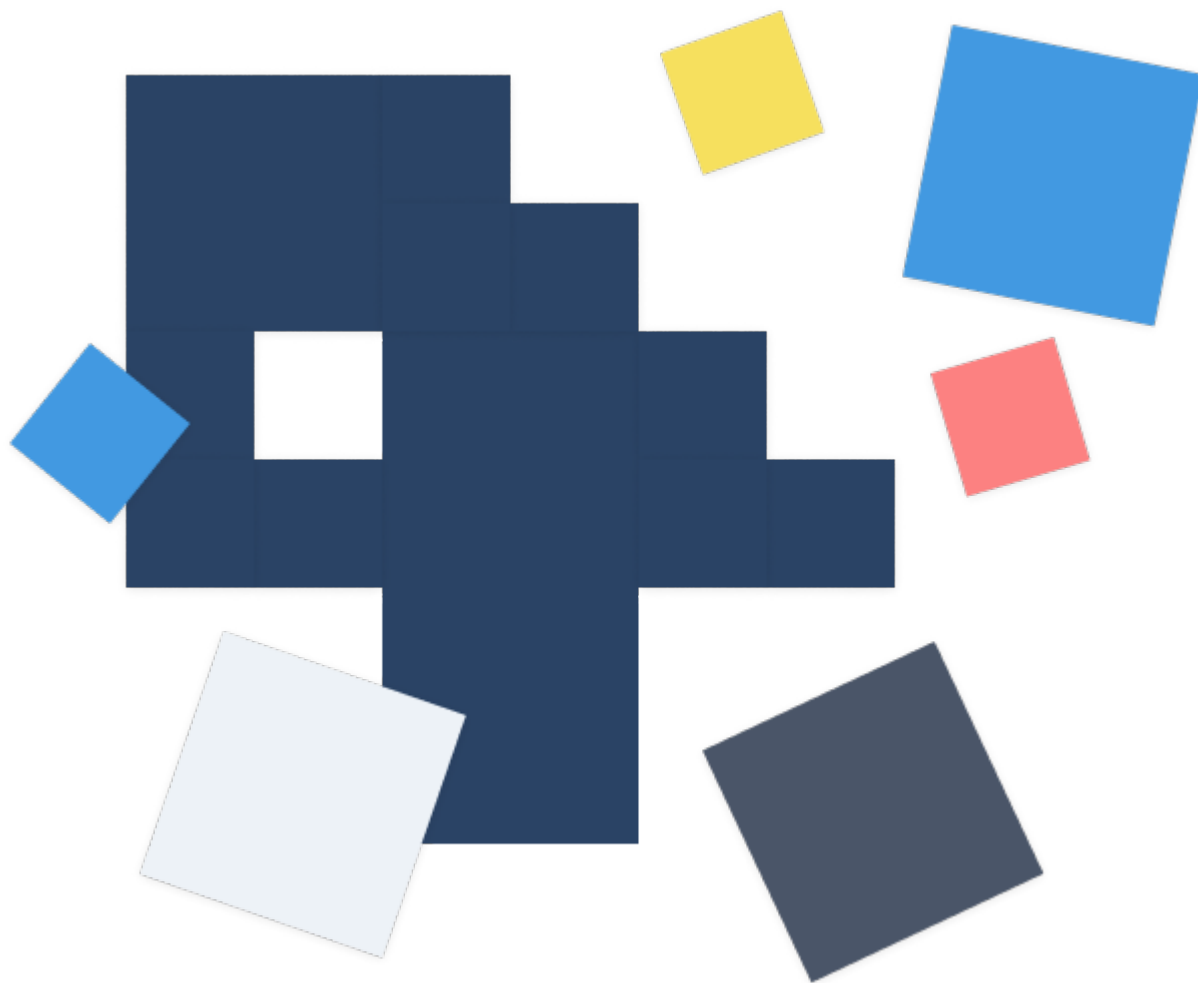
Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at <http://www.oracle.com/investor>. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

Introduction

- I am Peter (@pjausovec)
- Software Engineer at Oracle
- Working on "cloud-native" stuff
- Books:
 - [Cloud Native: Using Containers, Functions, and Data to Build Next-Gen Apps](#)
 - SharePoint Development
 - VSTO For Dummies
- Courses:
 - Kubernetes Course (<https://startkubernetes.com>)
 - Istio Service Mesh Course (<https://learnistio.com>)



MONOLITH



Microservices

Independently deployable services, owning their data and having well-defined interfaces

CODE

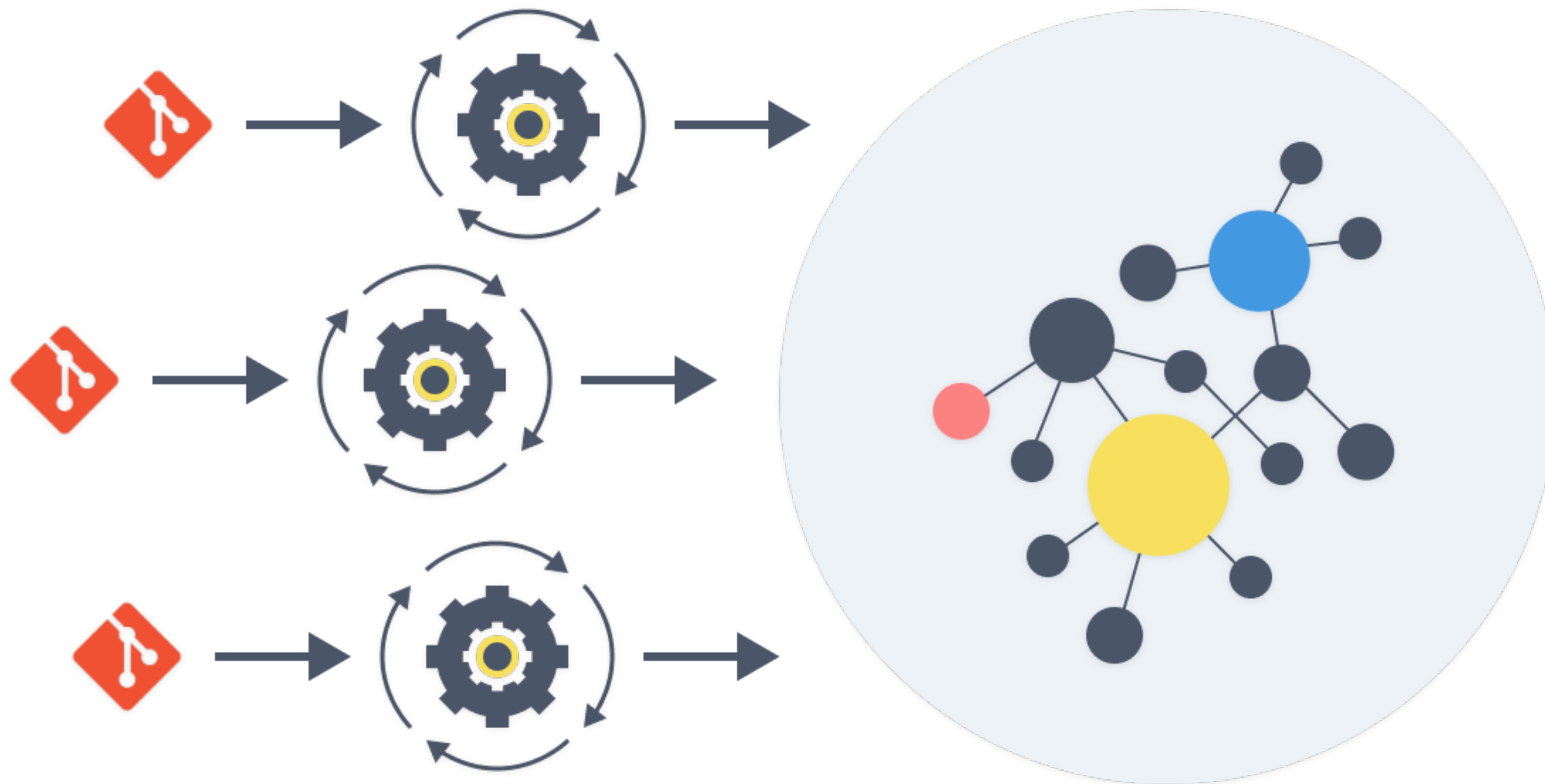


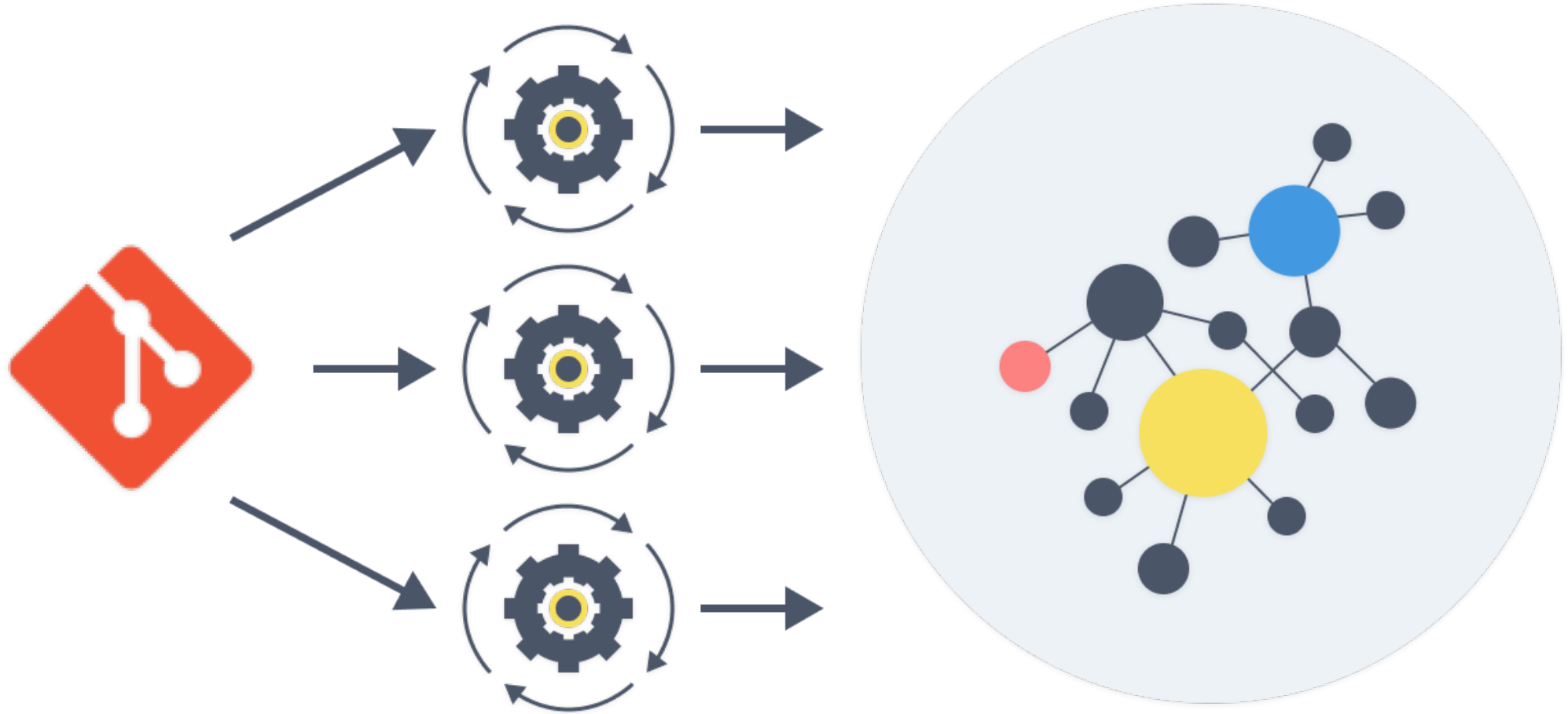
CI/CD



PRODUCTION







Service Resiliency

Resiliency

Ability to recover from failures and continue to function

Goal

Return the service to a fully functioning state after failure

Resiliency - High availability (1/2)

- Healthy 
- No significant downtime 
- Responsive 
- Meeting SLAs/SLOs 

Resiliency - Disaster Recovery (2/2)

- How to recover from incidents
 - Data backup and archiving
- DR starts when HA design can't handle the impact of failures

Path to resiliency

- Understand the requirements
- Define service availability
- Design for resiliency
- Strategies for detection and recovery
- Testing
- Monitoring

Path to resiliency

Understand the requirements

- How much downtime can you handle/is acceptable?
- More downtime → broken SLAs/SLOs → 🥲

Define service availability

- What does it mean for service to be available?

Design for resiliency

- Identify failures points - what and how can things wrong?

Path to resiliency

Strategies for detection and recovery

- How are you detecting and recovering from failures?

Testing and monitoring

- Test for failure conditions so you can detect and recover from them
- Monitor your services, so you know what's happening

Resiliency Strategies

- Load Balancing
- Timeouts and retries
- Circuit breakers and bulkhead pattern
- Data replication
- Graceful degradation
- Rate limiting

Load Balancing

Scale services by adding more instances

Timeouts and Retries

Timeouts

- Network latency: how long do you wait for responses?
- Waiting indefinitely == bad
- Always define timeouts!

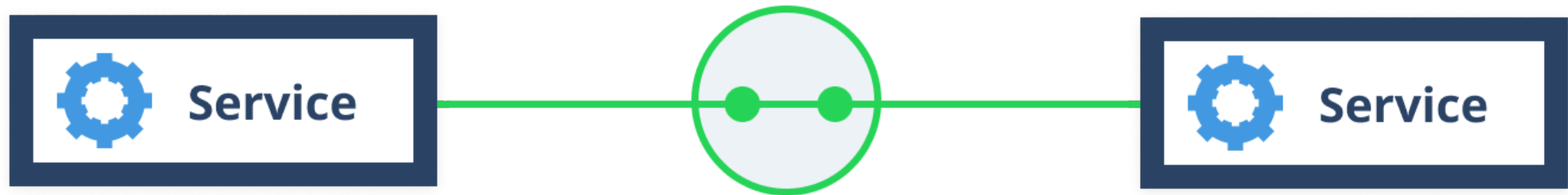
Retries

- Helps handle transient network failures
- Only retry calls that make sense
 - Idempotent operations
- Consider appropriate retry counts and intervals between retries

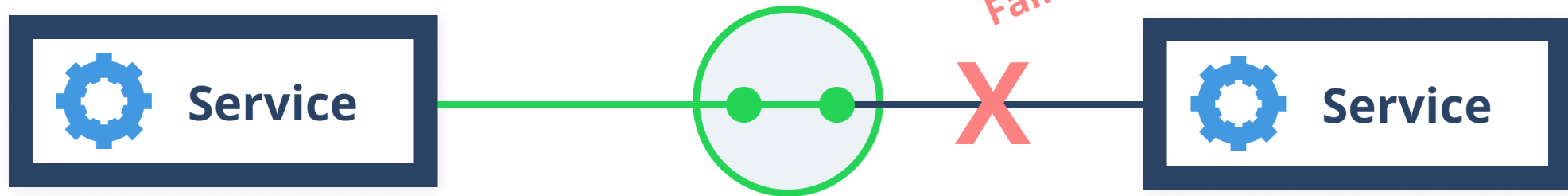
Circuit breaker

Prevents doing an operation that is likely to fail

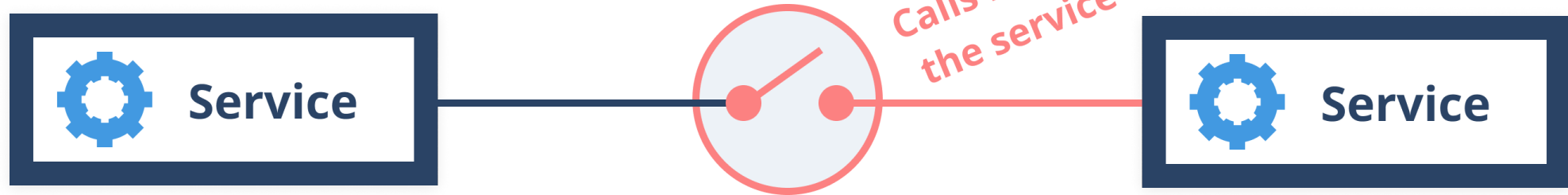
Circuit breaker



Circuit breaker

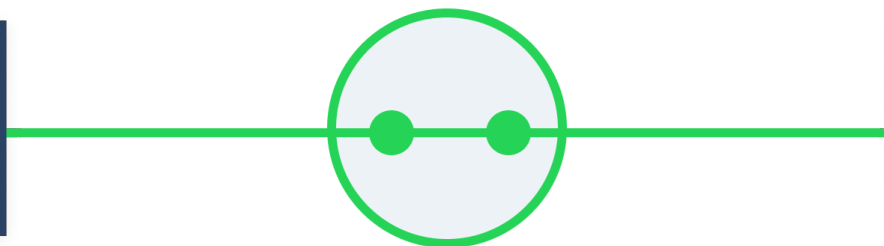


Circuit breaker





Circuit breaker



Bulkhead pattern

Isolate resources in such a way that if one fails, it's not affecting others

Data replication

Handle non-transient failures in the data store

Graceful degradation

Ability to maintain limited functionality in face of failures

Rate limiting

Restrict the number of requests made in a period of time

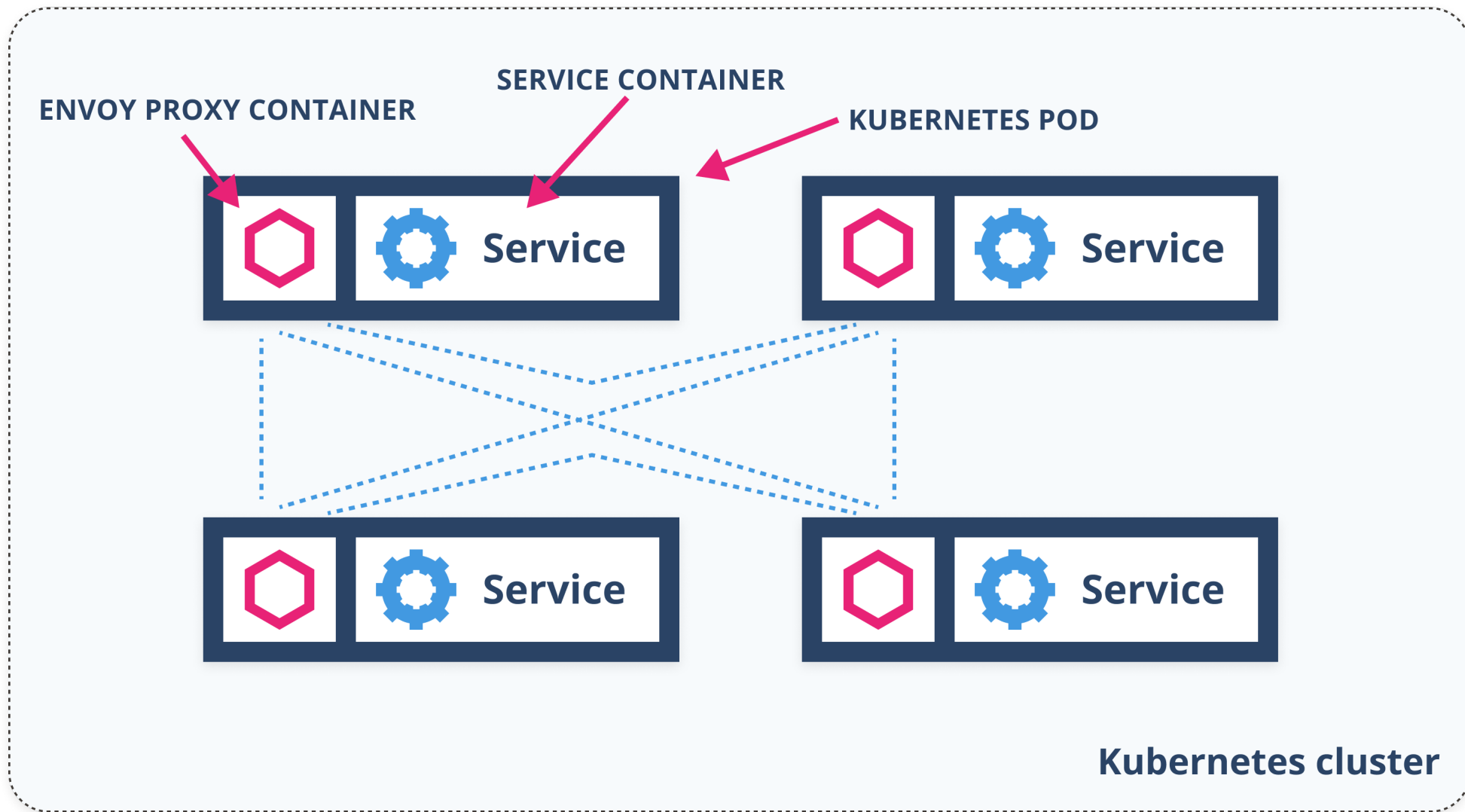
How?

Service Mesh

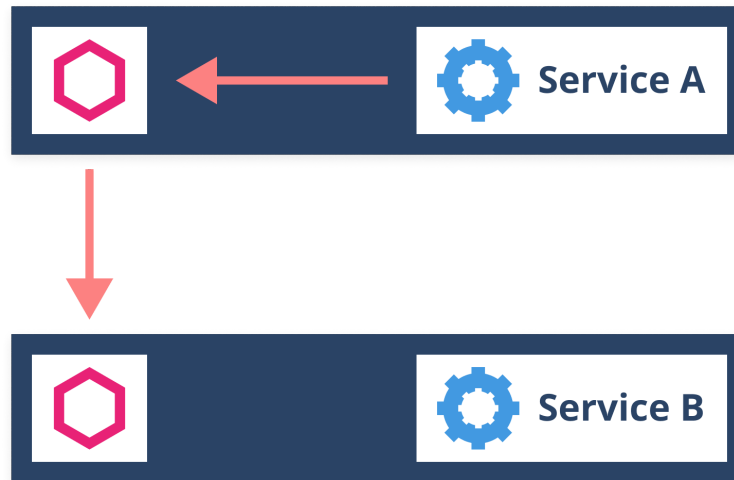
Dedicated infrastructure layer to connect, manage, and secure workloads by managing the communication between them

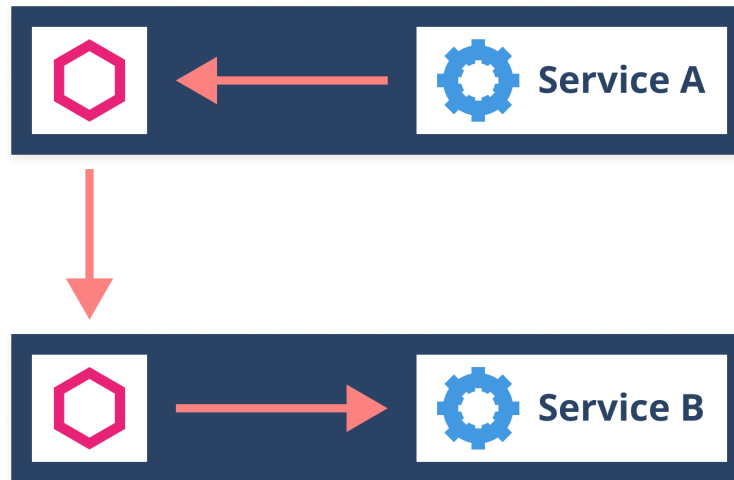
Istio service mesh

- Open source service mesh
 - Google, IBM, Lyft
- Well-defined API
- Can be deployed on-premise, in the cloud
 - Kubernetes
 - Mesos











Service Mesh & Resiliency

Resiliency Strategies - Service Mesh

- Load Balancing
- Timeouts and retries
- Circuit breakers and bulkhead pattern
- Data replication
- Graceful degradation
- Rate limiting



Testing for resiliency

- Test
- Measure
- Analyze (fix the issues)

Testing for Resiliency - Service Mesh

Inject failures

- Delays
 - Example: "For 30% of the requests, wait 5 seconds before responding"
- Faults
 - Example: "For 50% of the requests, return HTTP 404"

Failure injection

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: VirtualService
3 metadata:
4   name: service-b
5 spec:
6   hosts:
7   - service-b.default.svc.cluster.local
8   http:
9   - route:
10     - destination:
11       host: service-b.default.svc.cluster.local
12       subset: v1
13     fault:
14       abort:
15         percent: 60
16         httpStatus: 404
```



Delay injection

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: VirtualService
3 metadata:
4   name: service-b
5 spec:
6   hosts:
7   - service-b.default.svc.cluster.local
8   http:
9   - route:
10     - destination:
11       host: service-b.default.svc.cluster.local
12       subset: v1
13     fault:
14       delay:
15         percent: 20
16         fixedDelay: 3s
```



Timeouts & Retries

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: VirtualService
3 metadata:
4   name: service-b
5 spec:
6   hosts:
7     - service-b.default.svc.cluster.local
8   http:
9     - route:
10       - destination:
11         host: service-b.default.svc.cluster.local
12         subset: v1
13       timeout: 5s
14       retries:
15         attempts: 6
16         perTryTimeout: 2s
17       retryOn: gateway-error,connect-failure
```



Circuit Breakers

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: DestinationRule
3 metadata:
4   name: service-b
5 spec:
6   host: service-b.default.svc.cluster.local
7   trafficPolicy:
8     tcp:
9       maxConnections: 1
10    http:
11      http1MaxPendingRequests: 1
12      maxRequestsPerConnection: 1
13    outlierDetection:
14      consecutiveErrors: 1
15      interval: 1s
16      baseEjectionTime: 3m
17      maxEjectionPercent: 100
```



Observability

What is observability?

Act of measuring, collecting and analyzing metrics, traces, logs, events, ... from services

Logging

- Log granularity: verbose, debug, warning, info, error, ...
 - Storage is cheap - rather log more than less
- Store logs in a central place
- Use correlation IDs
- Don't log private information
- Use common format

Monitoring & Alerting

Monitoring

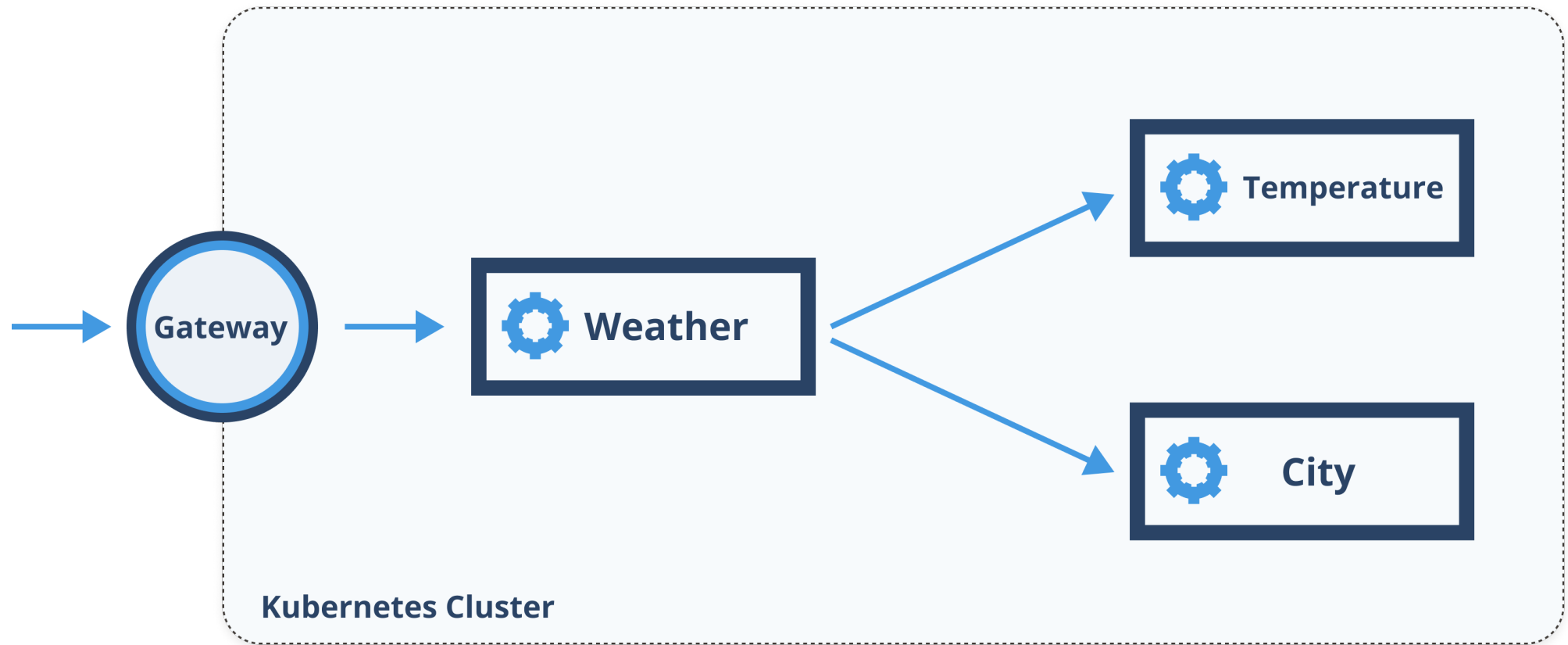
- Know the state of your system
- Collect system/infrastructure metrics (e.g. CPU, memory, ...) and business metrics (instrument services)

Alerting

- Queries that look for failures/failure conditions
- PagerDuty -> alert on-call engineers
- Only fire alerts when something is really wrong

Tools

- Grafana
- Jaeger
- Kiali
- ELK (Elastic Search + Fluentd + Kibana)
- PagerDuty



Resources

- Kubernetes on Oracle Cloud (OKE) - (<https://cloud.oracle.com>)
- Kubernetes - (<https://kubernetes.io>)
- Istio - (<https://istio.io>)

Oracle Microservices Example

- MuShop - <https://github.com/oracle-quickstart/oci-cloudnative>

Thank you

Slides: <https://slides.peterj.dev>

Contact

- [@pjausovec](https://twitter.com/pjausovec)
- <https://peterj.dev>

Table of Contents

- [Introduction](#)
- [Service Resiliency](#)
- [Service Mesh](#)
- [Service Mesh & Resiliency](#)
- [Resources](#)

