Advanced Application Management Using Red Hat OpenShift Service Mesh

Resiliency

Module Topics

- Resiliency
- Resiliency patterns
- Cascading failures
- Timeout, Circuit Breaker, Bounded Queue
- Service Mesh and Resiliency

Resiliency

- Ability of system to gracefully handle and recover from failure
  - Best case: Without user knowing it
  - Worst case: With graceful degradation of service
- Distributed applications have increased probability of something going wrong
  - Best approach: Embrace failures!
- Favor MTTR over MTTF
  - MTTR: Mean time to recovery
  - MTTF: Mean time to failure

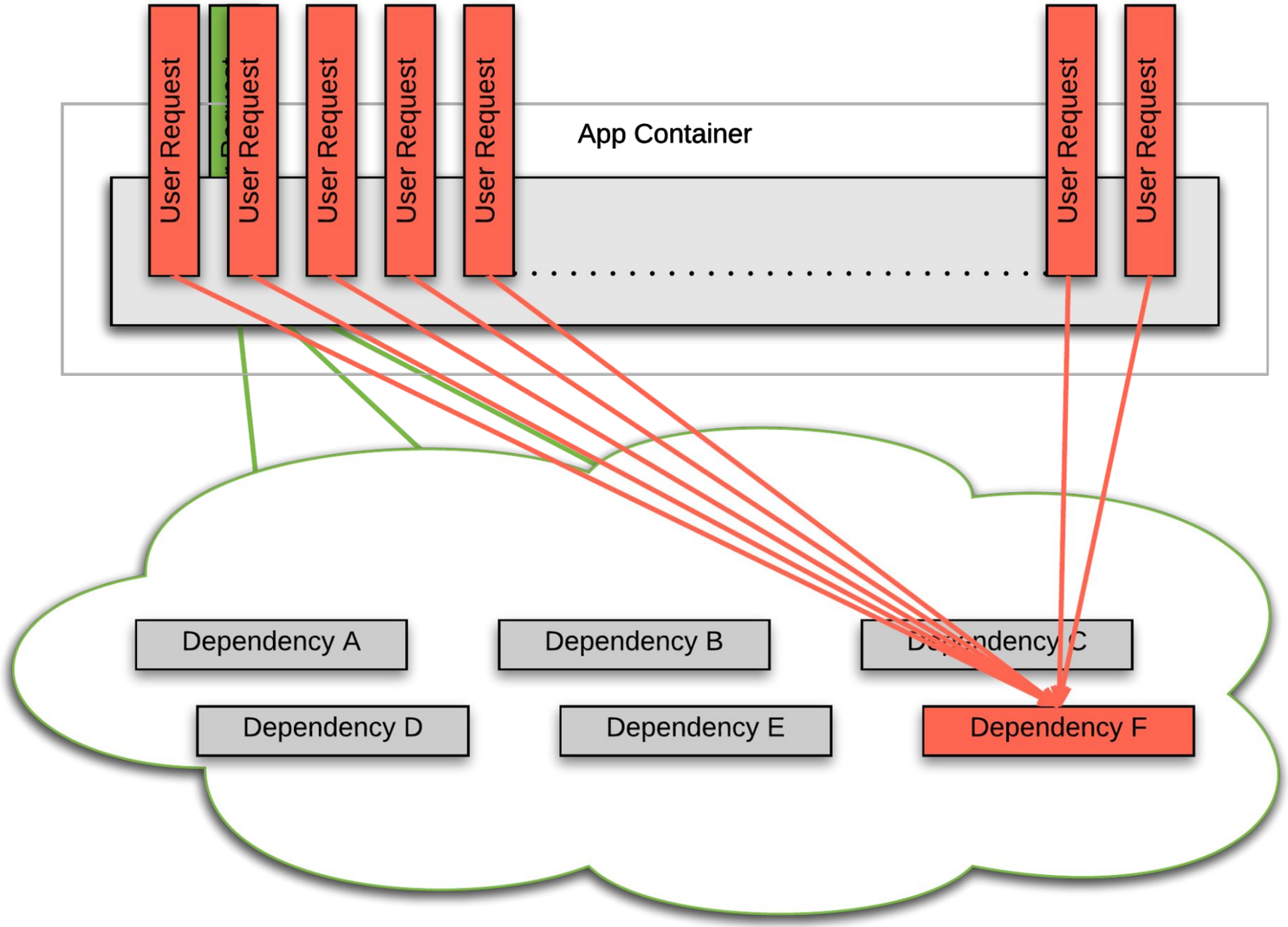
Resiliency Patterns

- Isolation
  - Bulkhead
  - Parameter checking
  - Health endpoint monitoring
- Loose coupling
  - Statelessness
  - Idempotency
  - Asynchronous communication
  - Location transparency
- Latency control
  - Timeout
  - Fail fast
  - Circuit breaker
  - Bounded queue

Cascading Failures

- Complete system fails as result of failing component
- Example:
  - Application running in app container with dependencies to N subsystems
  - Single dependency shows increased latency under high volume
  - User-request threads become saturated
  - Entire application becomes unresponsive

Cascading Failures



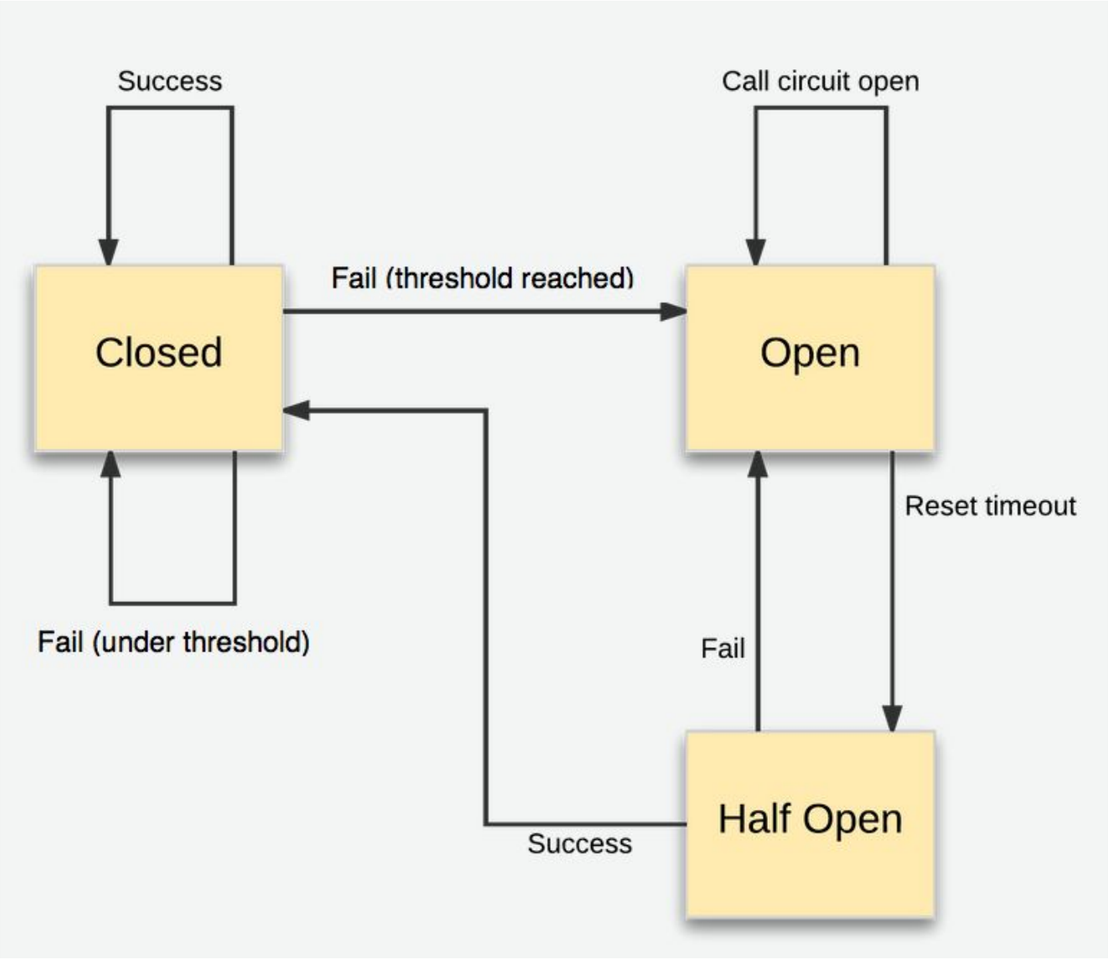
Timeout

- Preserve responsiveness independent of upstream latency
- Measure response time of upstream service
- Stop waiting after predetermined time
- Take alternative action if timeout is reached

Circuit Breaker

- Circuit breaker wraps calls to dependent services
- Original state: Closed
- Circuit breaker implementation monitors calls to dependent service
- When failures (timeout, exception) hit preconfigured threshold, circuit breaker trips open
- When open, calls to dependent service no longer made, fallback response returned
- Circuit breaker moves to half-open state after configurable time
- In half-open state, circuit breaker executes remote calls periodically to monitor dependent service's health
- When service becomes healthy again, state switches back to closed

Circuit Breaker



Bounded Queue

- Limit request queue sizes in front of highly utilized resources
- Avoids latency due to overloaded resources
- Introduces pushback on client (back pressure)

Service Mesh and Resiliency

- Client-side load balancing
- Circuit breaker: Outlier detection
- Automatic retry
- Request timeout handling
- Fault injection (chaos engineering)

Load Balancing

- Configured through DestinationRule
- Round-robin by default
- Also supports:
  - Random
  - Weighted
  - Fewest requests
  - Consistent hash: Based on header, cookie, source IP

```
apiVersion:networking.istio.io/v1alpha3
kind:DestinationRule
metadata:
  name:foo-default
spec:
  host:foo.default.svc.cluster.local
  trafficPolicy:
    loadBalancer:
      simple:LEAST_CONN
```

Outlier Detection

- Istio implementation of circuit breaker
- Outliers excluded from active load-balancing pool

```
apiVersion:networking.istio.io/v1alpha3
kind:DestinationRule
metadata:
  name:foo-default
spec:
  host:foo.default.svc.cluster.local
  trafficPolicy:
    outlierDetection:
      consecutiveErrors:5
      interval:1m
      baseEjectionTime:3m
```

Retries and Timeouts

- Default timeout in Envoy proxy: 15s
- Automatic retry when error code 503 returned
- Defined on per-service basis using VirtualService

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - route:
    - destination:
        host: ratings
        subset: v1
    timeout: 10s
    retries:
      attempts: 3
      perTryTimeout: 2s
```

Fault Injection

- Chaos engineering
- Test distributed applications by deliberately injecting faults into running (production) systems
- Inject delays and HTTP response error codes at application layer
- Defined on per-service basis using VirtualService

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        percentage:
          value:
        fixedDelay: 5s
```

Module Summary

- Resiliency
- Resiliency patterns
- Cascading failures
- Timeout, Circuit Breaker, Bounded Queue
- Service Mesh and Resiliency