Designing Apps for Resiliency

Masashi Narumoto

Principal Lead PM

AzureCAT patterns & practices

Agenda

- What is ’resiliency’?
- Why it’s so important?
- Process to improve resiliency
- Resiliency checklist

What is ‘Resiliency’?

- **Resiliency** is the ability to recover from failures and continue to function. It's not about *avoiding* failures, but *responding* to failures in a way that avoids downtime or data loss.

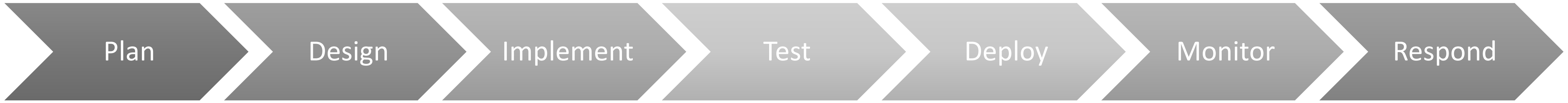
- **High availability** is the ability of the application to keep running in a healthy state, without significant downtime.

- **Disaster recovery** is the ability to recover from rare but major incidents: Non-transient, wide-scale failures, such as service disruption that affects an entire region.

Why it’s so important?

- More transient faults in the cloud
- Dependent service may go down
- SLA < 100% means something could go wrong at some point
- More focus on MTTR rather than MTBF

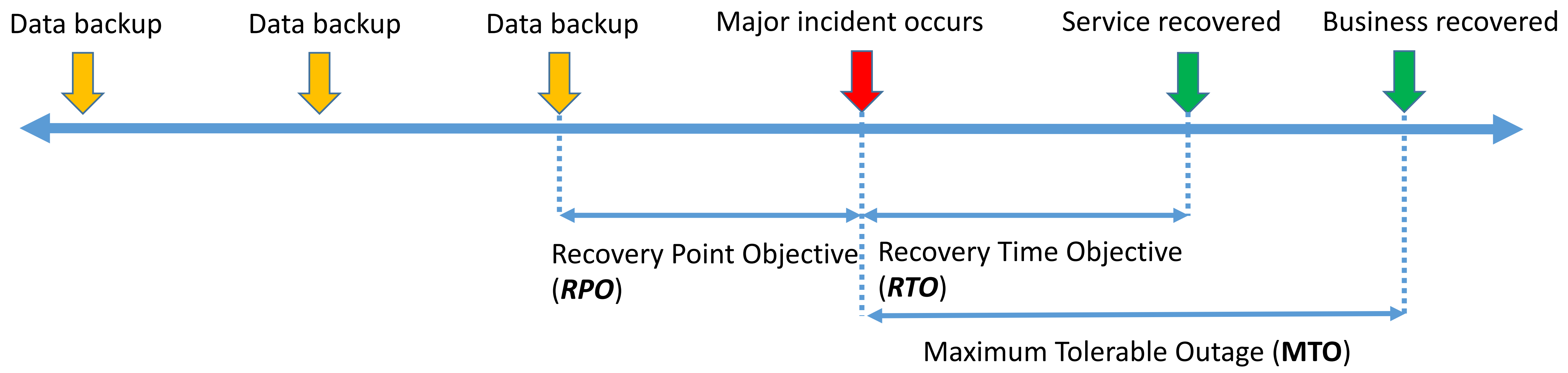
Process to improve resiliency



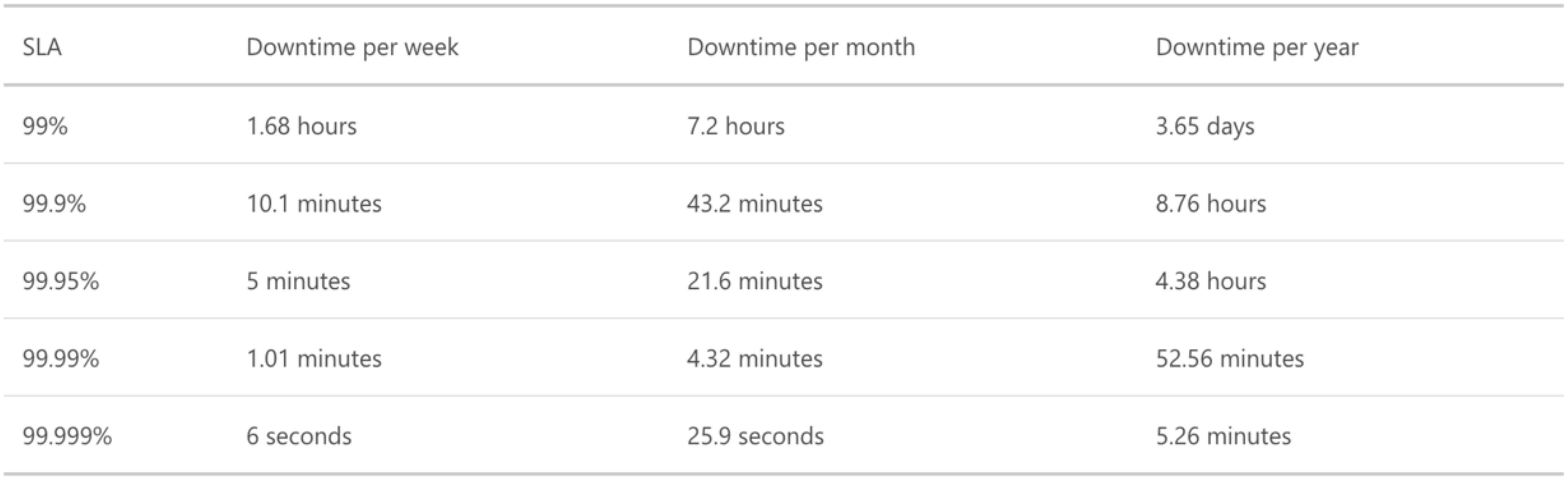
Define requirements	Identify failures	Implement recovery strategies	Inject failures Simulate FO	Deploy apps in a reliable manner	Monitor failures	Take actions to fix issues

Defining resiliency requirements

RPO: The maximum time period in which data might be lost
RTO: Duration of time in which the service must be restored after an incident


Data backup
Data backup
Data backup
Major incident occurs
Service recovered
Business recovered
Recovery Point Objective (*RPO*)
Recovery Time Objective (*RTO*)
Maximum Tolerable Outage (**MTO**)

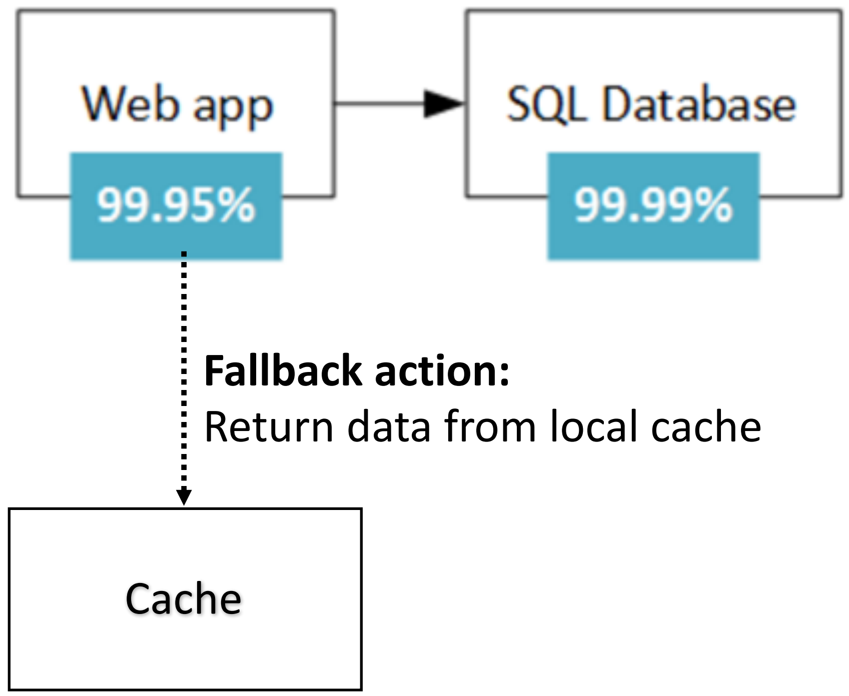
SLA (Service Level Agreement)



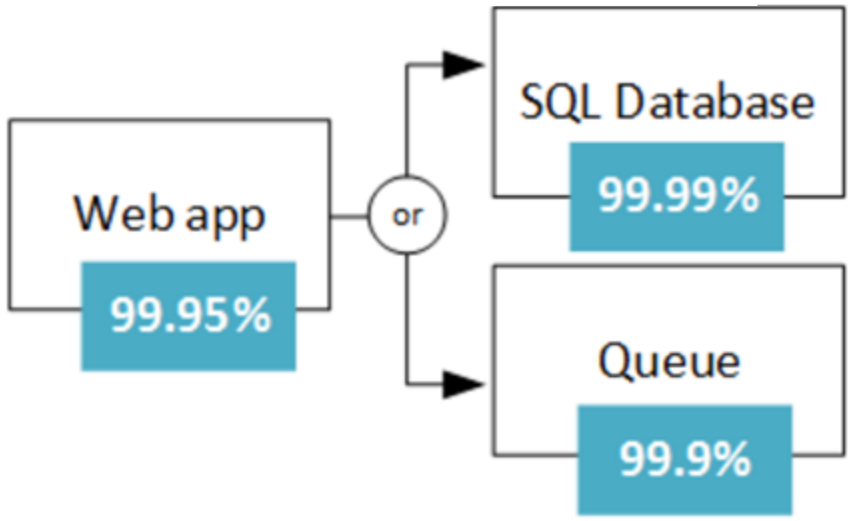
SLA	Downtime per week	Downtime per month	Downtime per year
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10.1 minutes	43.2 minutes	8.76 hours
99.95%	5 minutes	21.6 minutes	4.38 hours
99.99%	1.01 minutes	4.32 minutes	52.56 minutes
99.999%	6 seconds	25.9 seconds	5.26 minutes

Composite SLA

Composite SLA = ~~99.95% x 99.99% = 99.94%~~ **99.95%**



Composite SLA = **99.95%**



1.0 − (0.0001 × 0.001) = 99.99999%

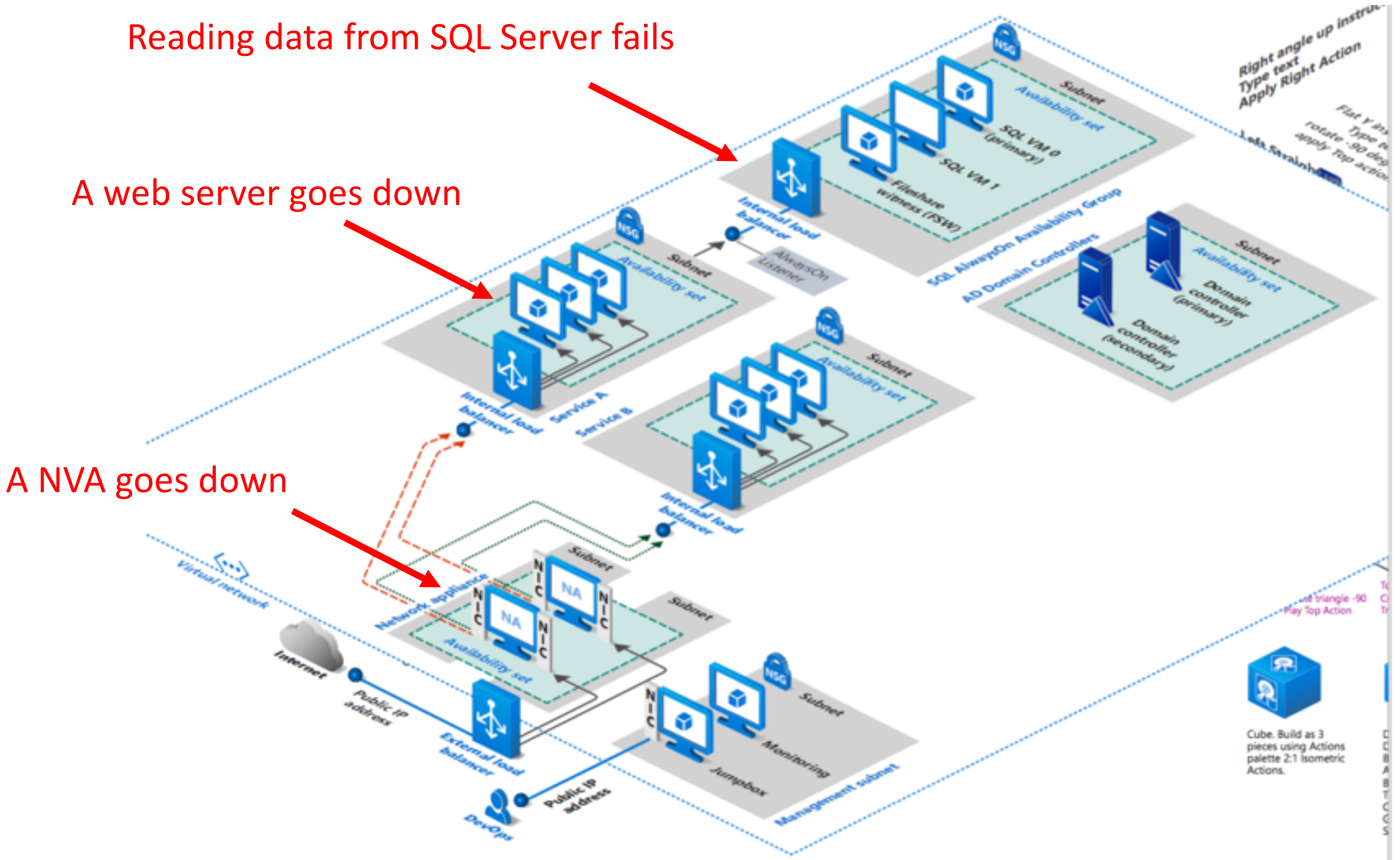
Composite SLA for two regions = (1 − (1 − N)(1 − N)) x Traffic manager SLA

(1 – (1 – 0.9995) x (1 – 0.9995)) x 0.9999 = 0.999899

patterns & practices
proven practices for predictable results

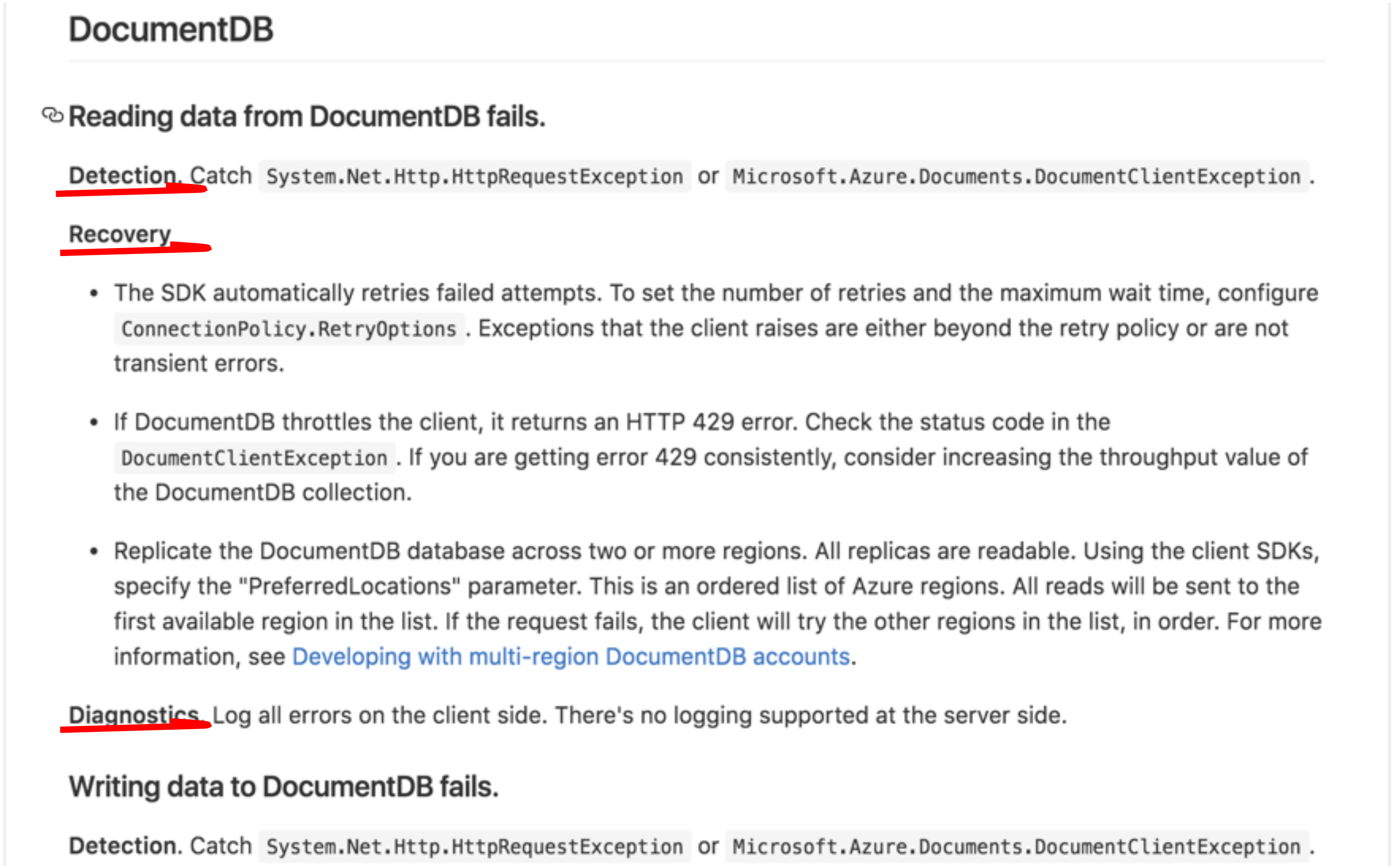
Designing for resiliency

1. Identify possible failures
2. Rate risk of each failure (impact x likelihood)
3. Design resiliency strategy
   - Detection
   - Recovery
   - Diagnostics



Failure mode analysis

https://azure.microsoft.com/en-us/documentation/articles/guidance-resiliency-failure-mode-analysis/



DocumentDB

Reading data from DocumentDB fails.

Detection. Catch `System.Net.Http.HttpRequestException` or `Microsoft.Azure.Documents.DocumentClientException`.

Recovery

- The SDK automatically retries failed attempts. To set the number of retries and the maximum wait time, configure `ConnectionPolicy.RetryOptions`. Exceptions that the client raises are either beyond the retry policy or are not transient errors.
- If DocumentDB throttles the client, it returns an HTTP 429 error. Check the status code in the `DocumentClientException`. If you are getting error 429 consistently, consider increasing the throughput value of the DocumentDB collection.
- Replicate the DocumentDB database across two or more regions. All replicas are readable. Using the client SDKs, specify the "PreferredLocations" parameter. This is an ordered list of Azure regions. All reads will be sent to the first available region in the list. If the request fails, the client will try the other regions in the list, in order. For more information, see Developing with multi-region DocumentDB accounts.

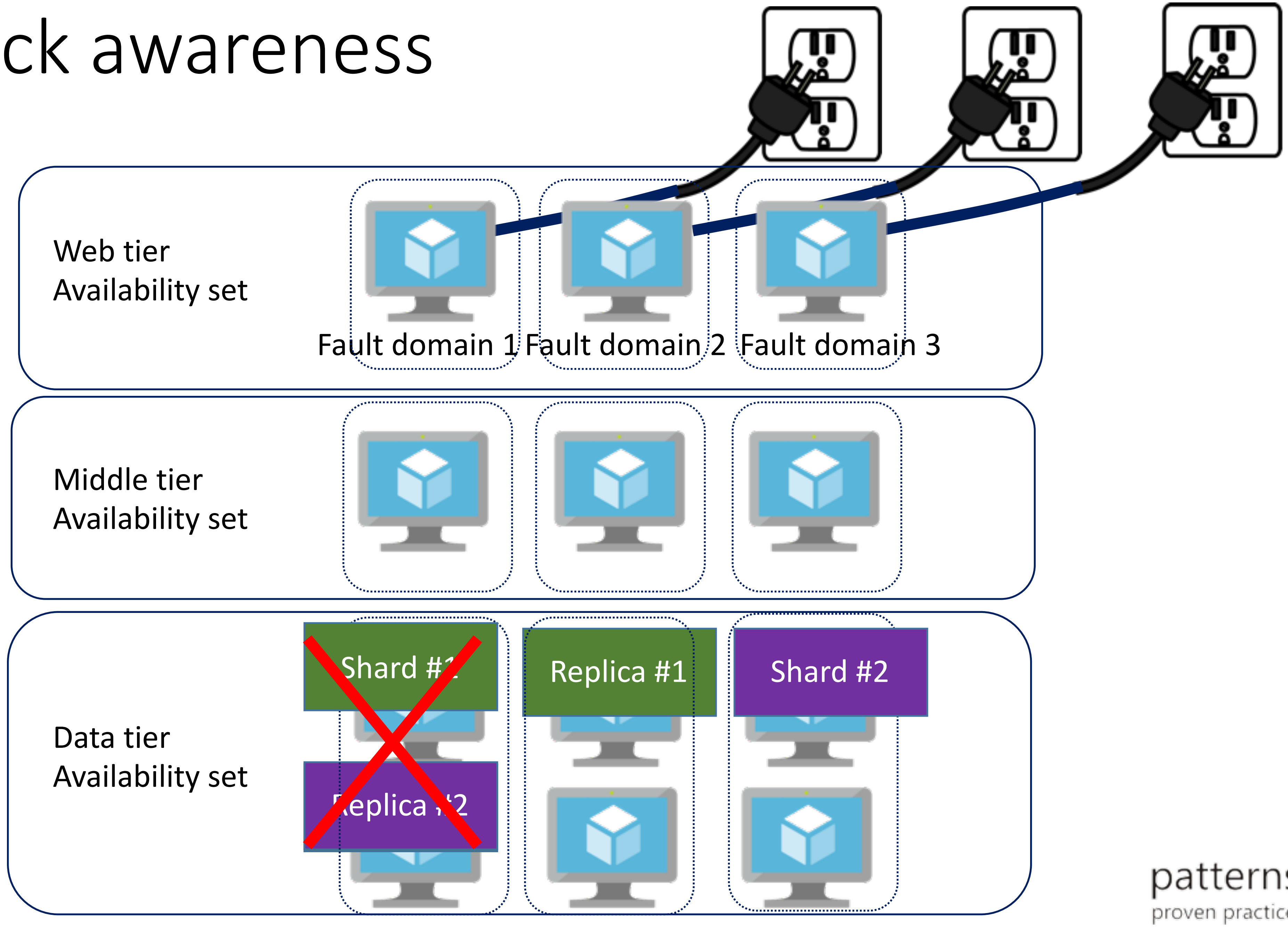
Diagnostics. Log all errors on the client side. There's no logging supported at the server side.

Writing data to DocumentDB fails.

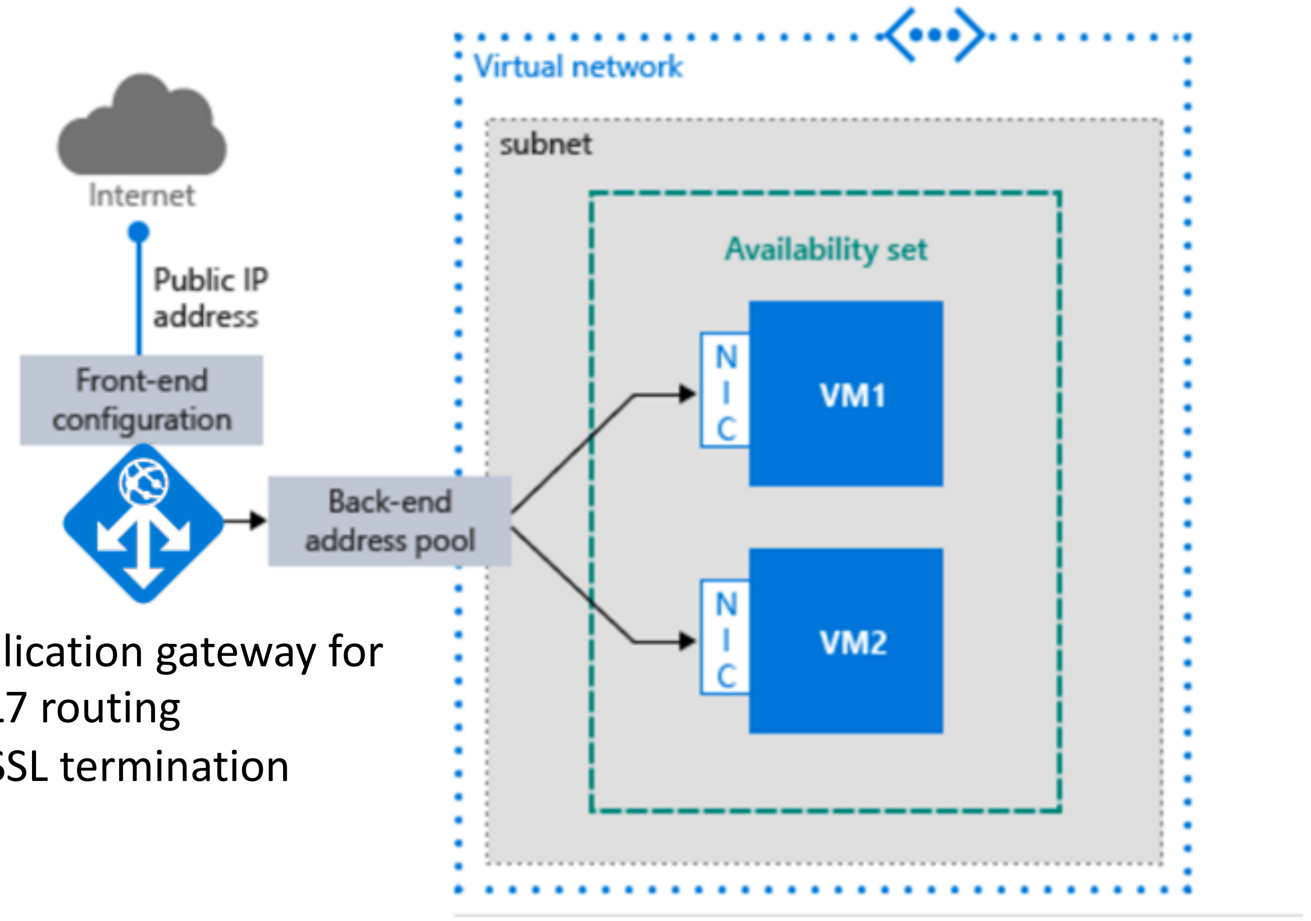
Detection. Catch `System.Net.Http.HttpRequestException` or `Microsoft.Azure.Documents.DocumentClientException`.

patterns & practices
proven practices for predictable results

Rack awareness

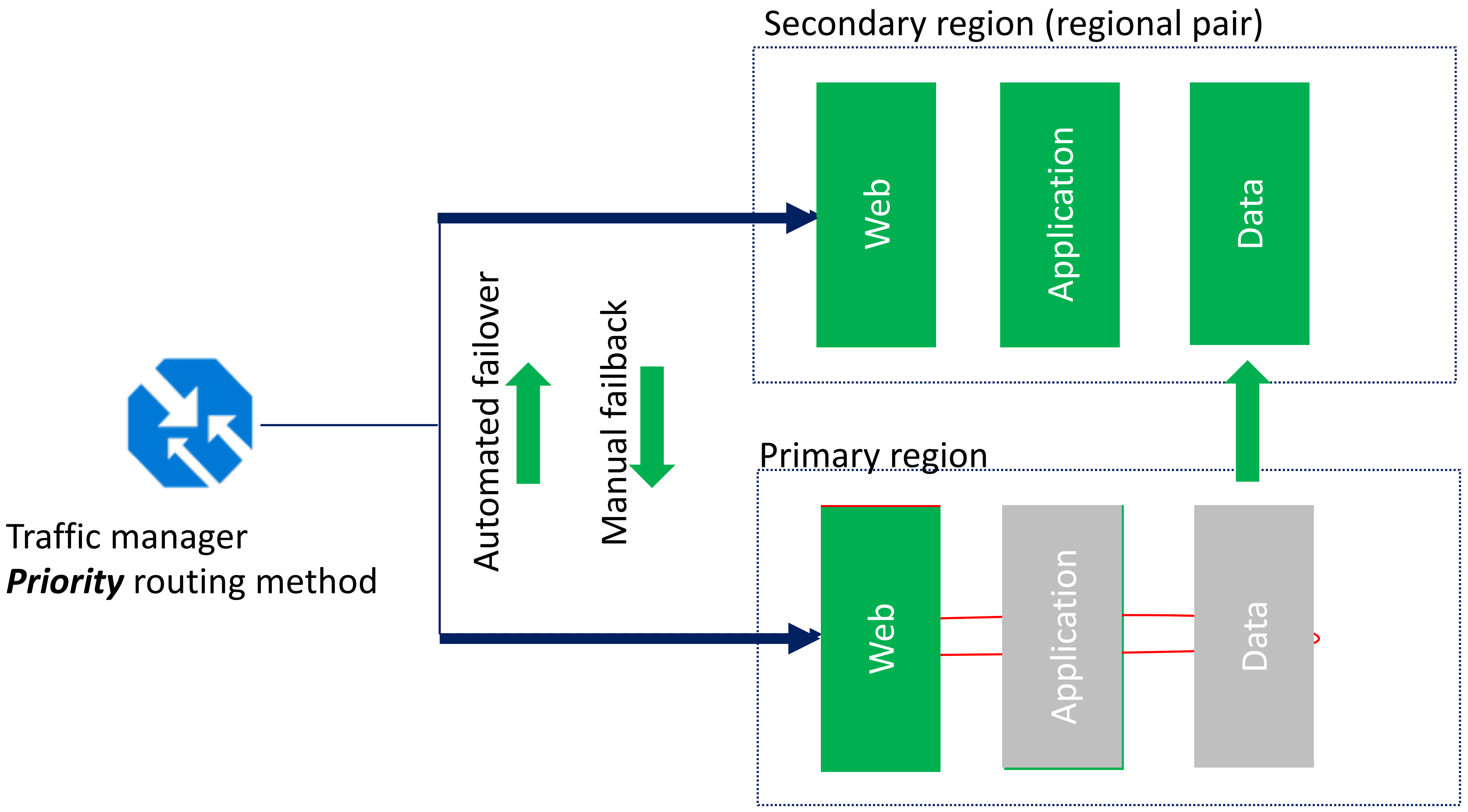


Load balance multiple instances

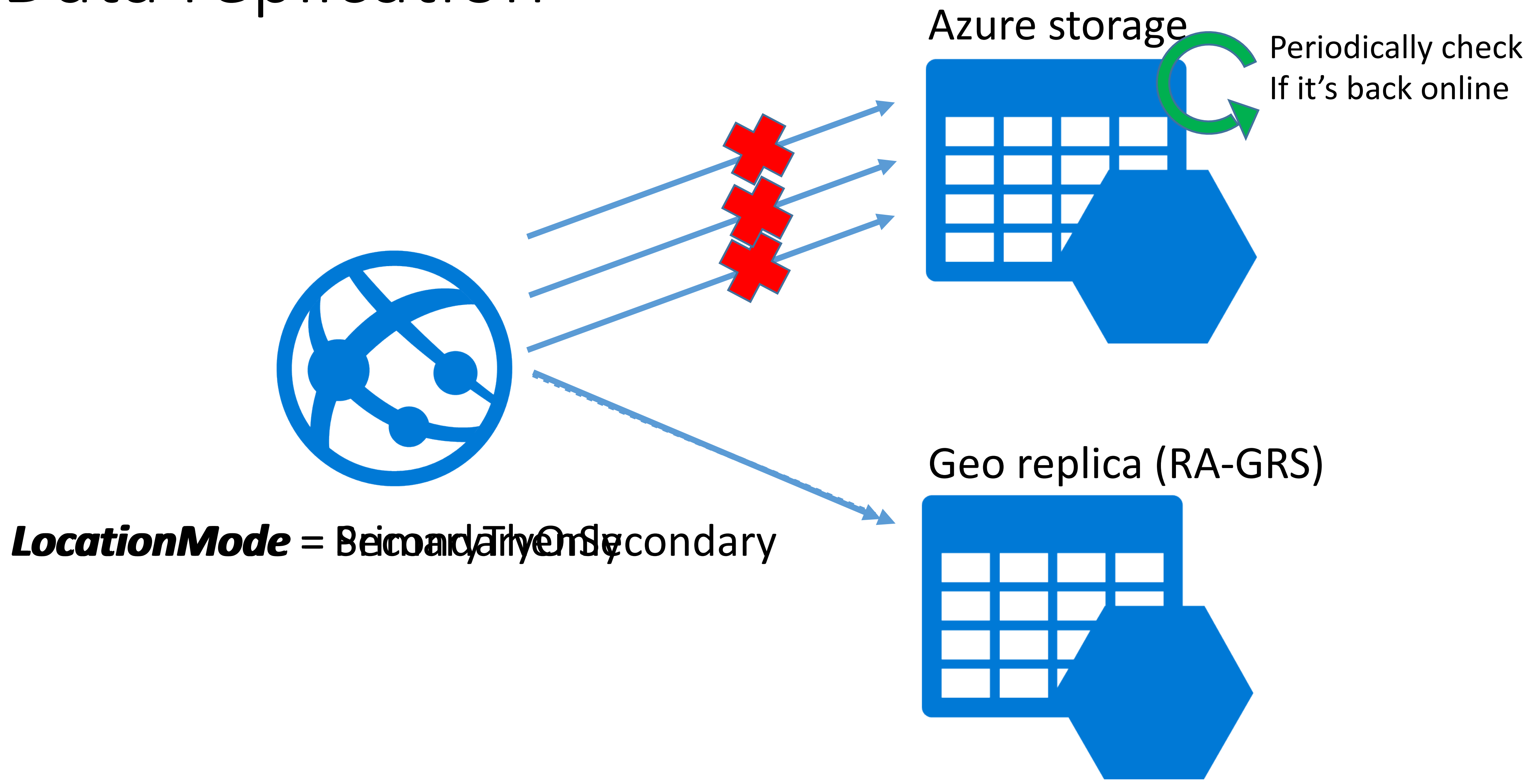


Application gateway for
- L7 routing
- SSL termination

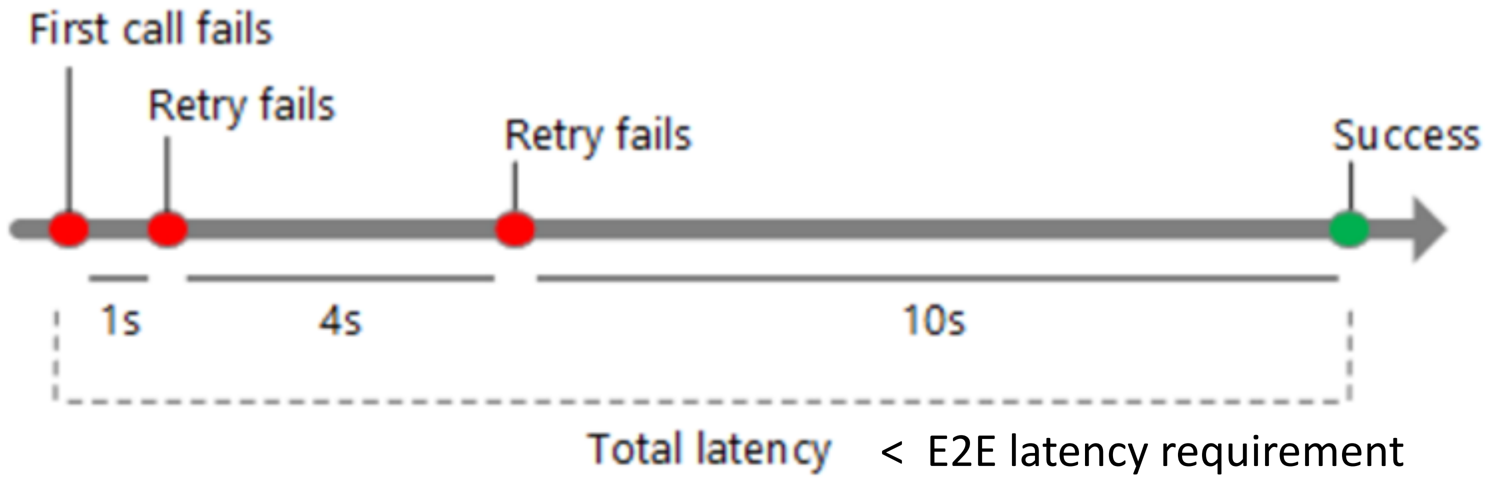
Failover / Failback



Data replication



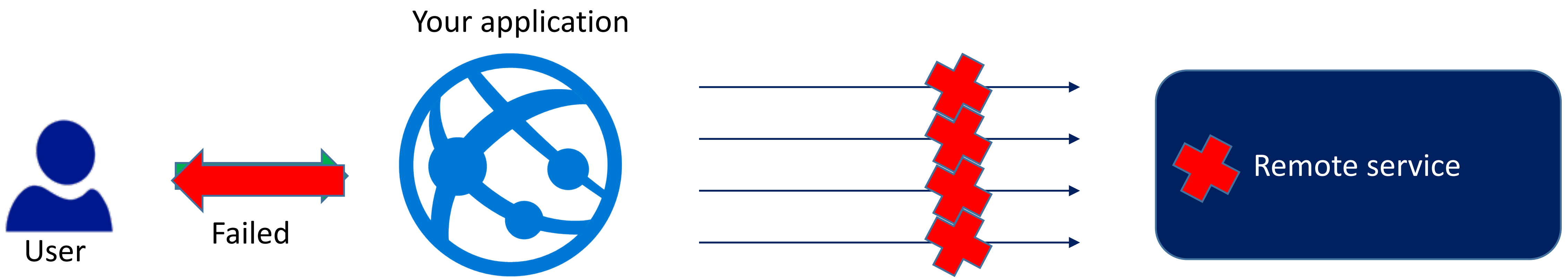
Retry transient failures



< E2E latency requirement

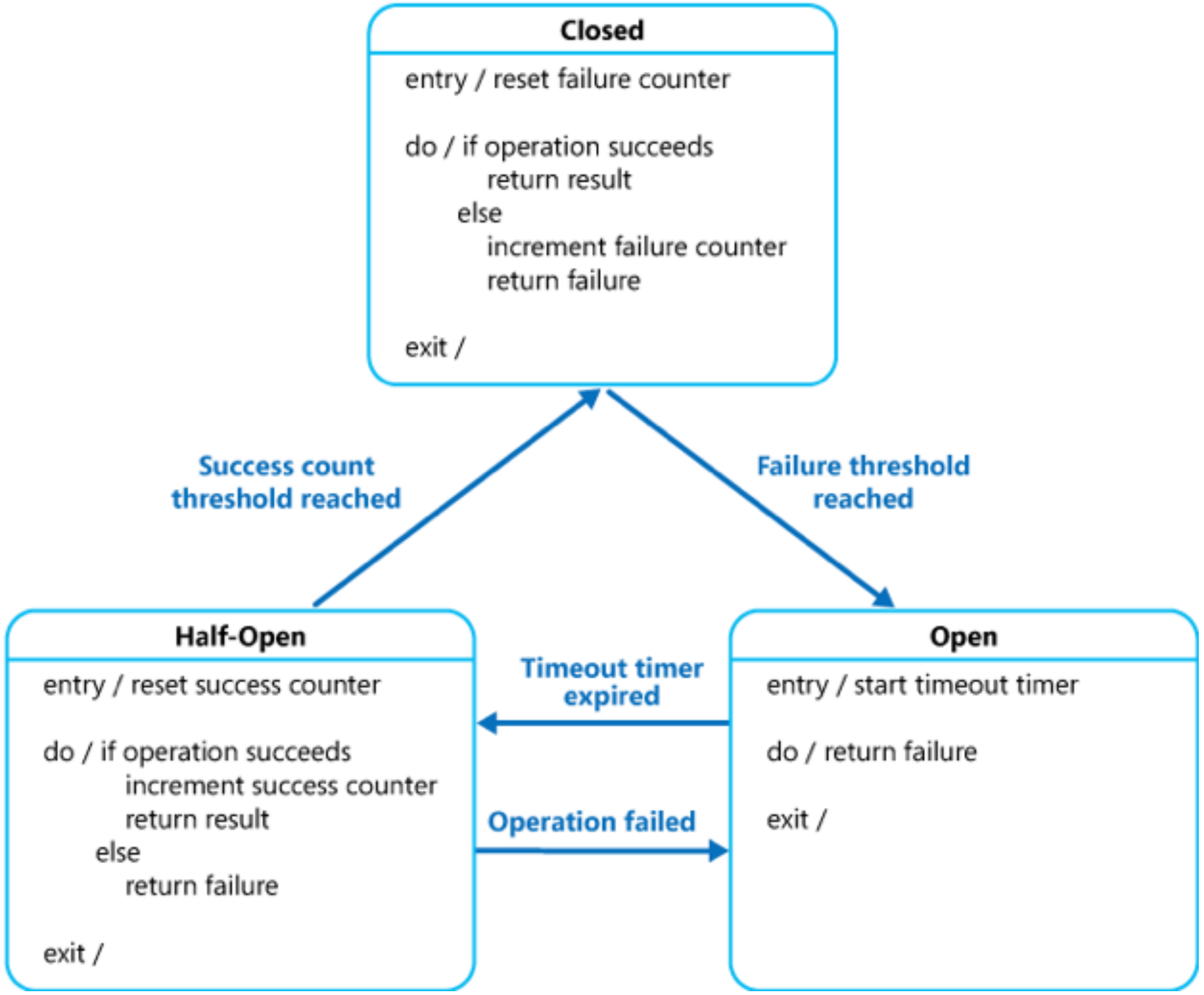
See 'Azure retry guidance' for more details

Circuit Breaker



Hold resources while retrying operation
Lead to cascading failures

Circuit Breaker



```
Closed
entry / reset failure counter

do / if operation succeeds
            return result
        else
            increment failure counter
            return failure

exit /
```

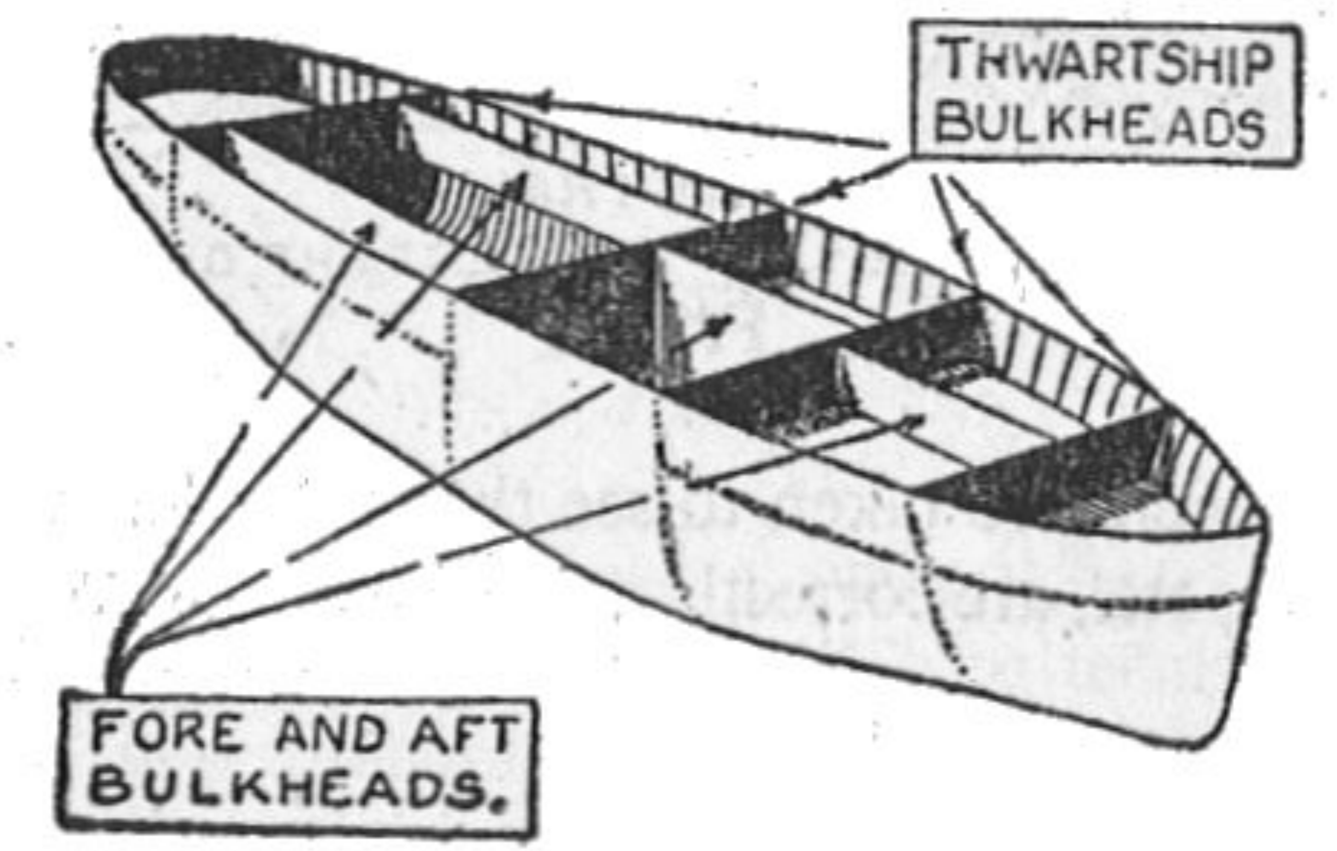
```
Half-Open
entry / reset success counter

do / if operation succeeds
            increment success counter
            return result
        else
            return failure

exit /
```

```
Open
entry / start timeout timer

do / return failure

exit /
```

https://github.com/App-vNext/Polly

patterns & practices
proven practices for predictable results

Bulkhead

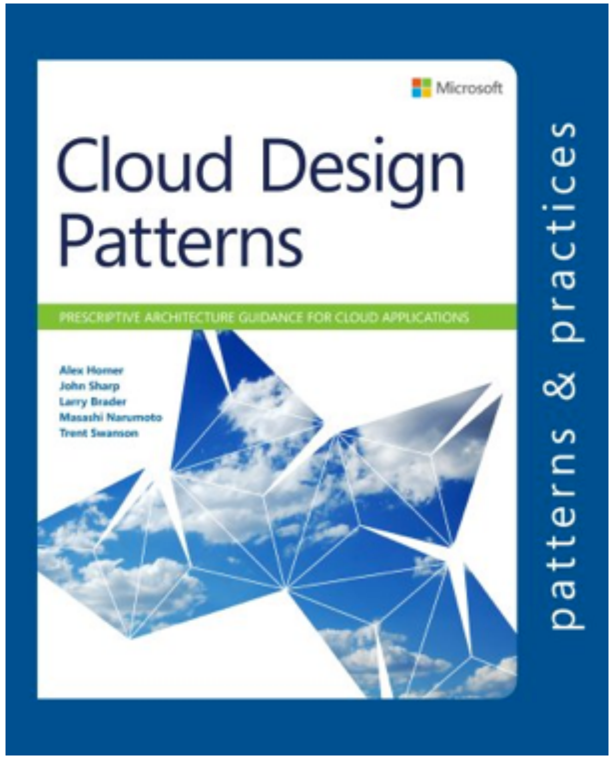


Memory
CPU
Disk
Thread pool
Connection pool
Network connection

Other design patterns for resiliency

- Compensating transaction
- Scheduler-agent-supervisor
- Throttling
- Load leveling
- Leader election

See ‘Cloud design patterns’



Microsoft

Cloud Design Patterns

PRESCRIPTIVE ARCHITECTURE GUIDANCE FOR CLOUD APPLICATIONS

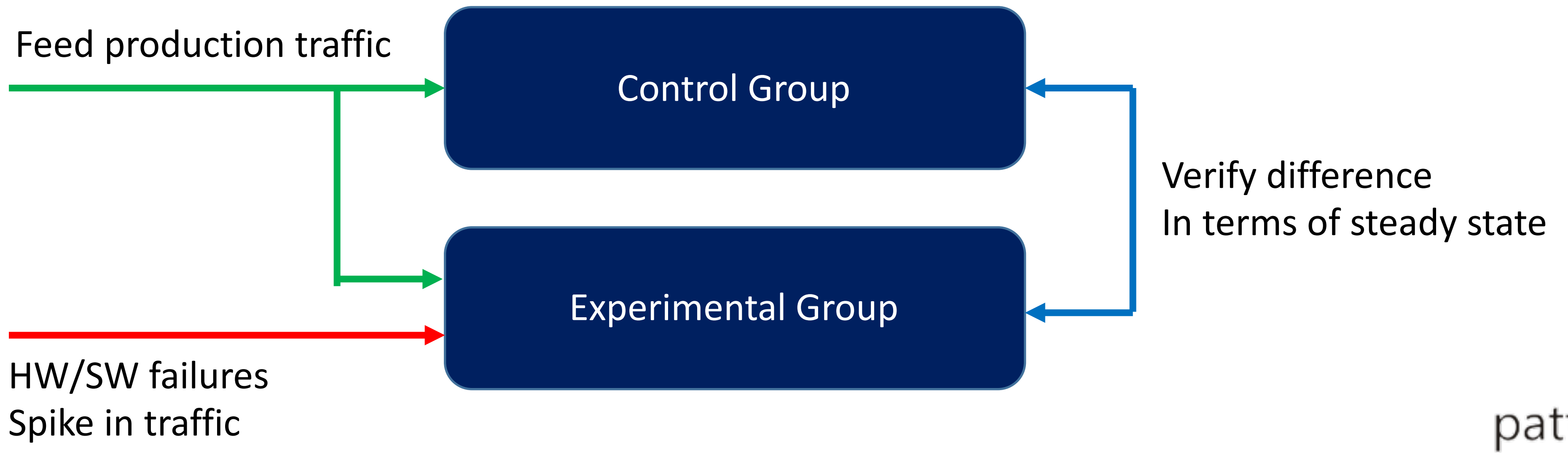
Alex Homer
John Sharp
Larry Brader
Masashi Narumoto
Trent Swanson

patterns & practices

Principles of chaos engineering

http://principlesofchaos.org/

- Build hypothesis around steady state behavior
- Vary real-world events
- Run experiments in production
- Automate experiments to run consistently

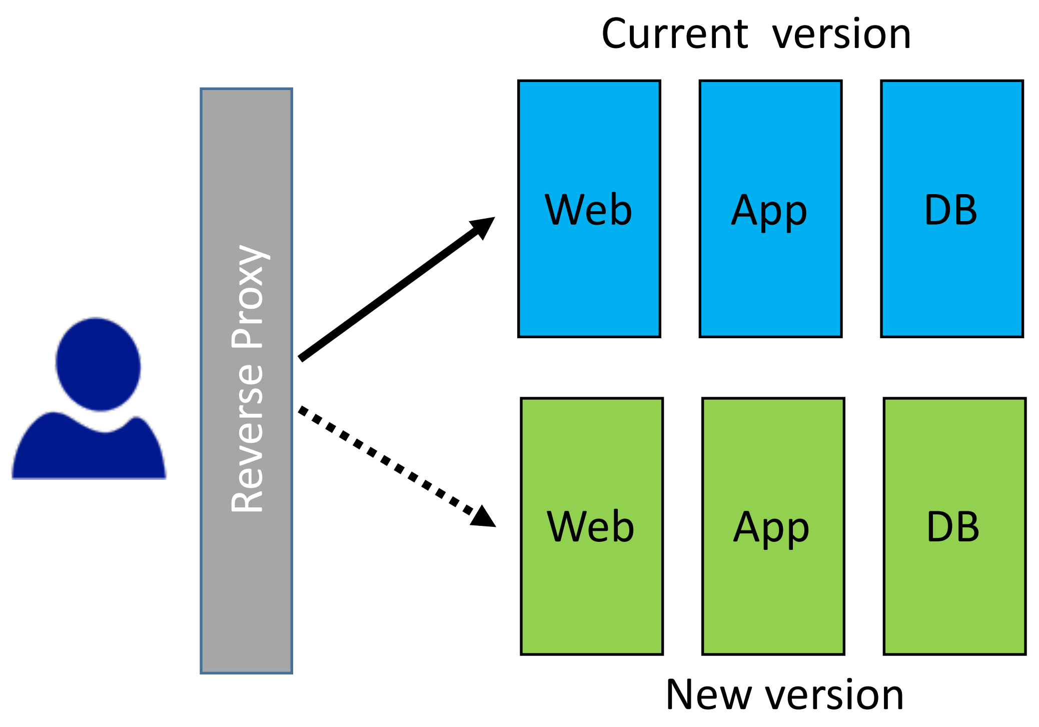


patterns & practices
proven practices for predictable results

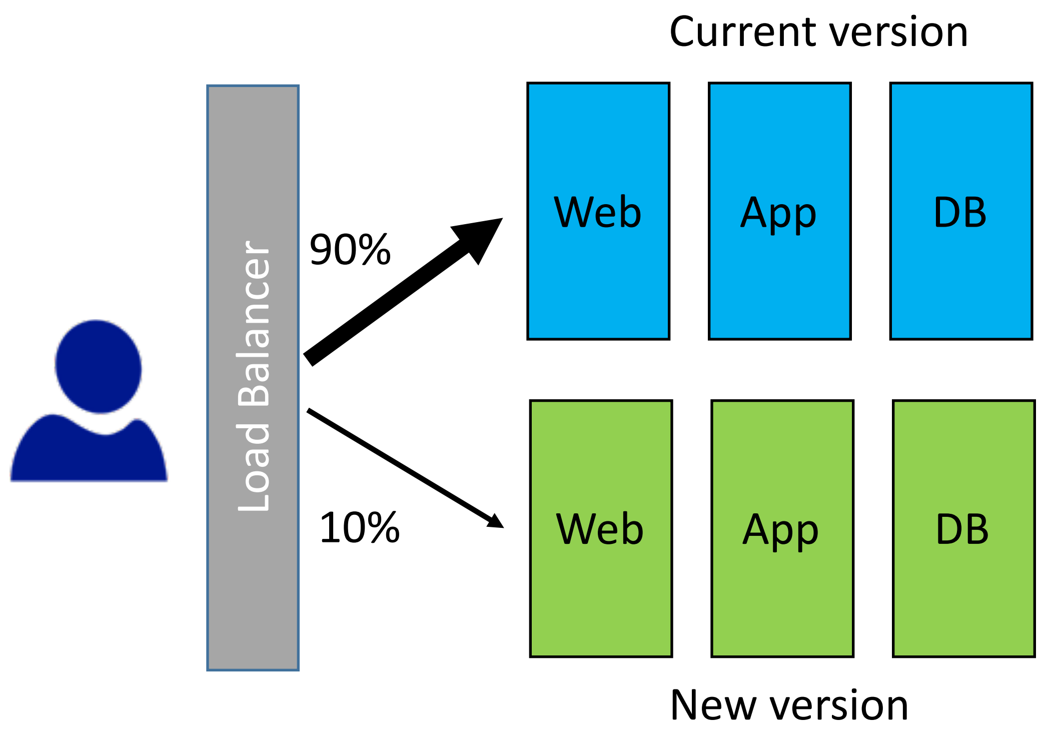
Testing for resiliency

- Fault injection testing
  - Shut down VM instances
  - Crash processes
  - Expire certificates
  - Change access keys
  - Shut down the DNS service on domain controllers
  - Limit available system resources, such as RAM or number of threads
  - Unmount disks
  - Redeploy a VM
- Load testing
  - Use production data as much you can
  - VSTS, JMeter
- Soak testing
  - Longer period under normal production load

Blue/Green and Canary release



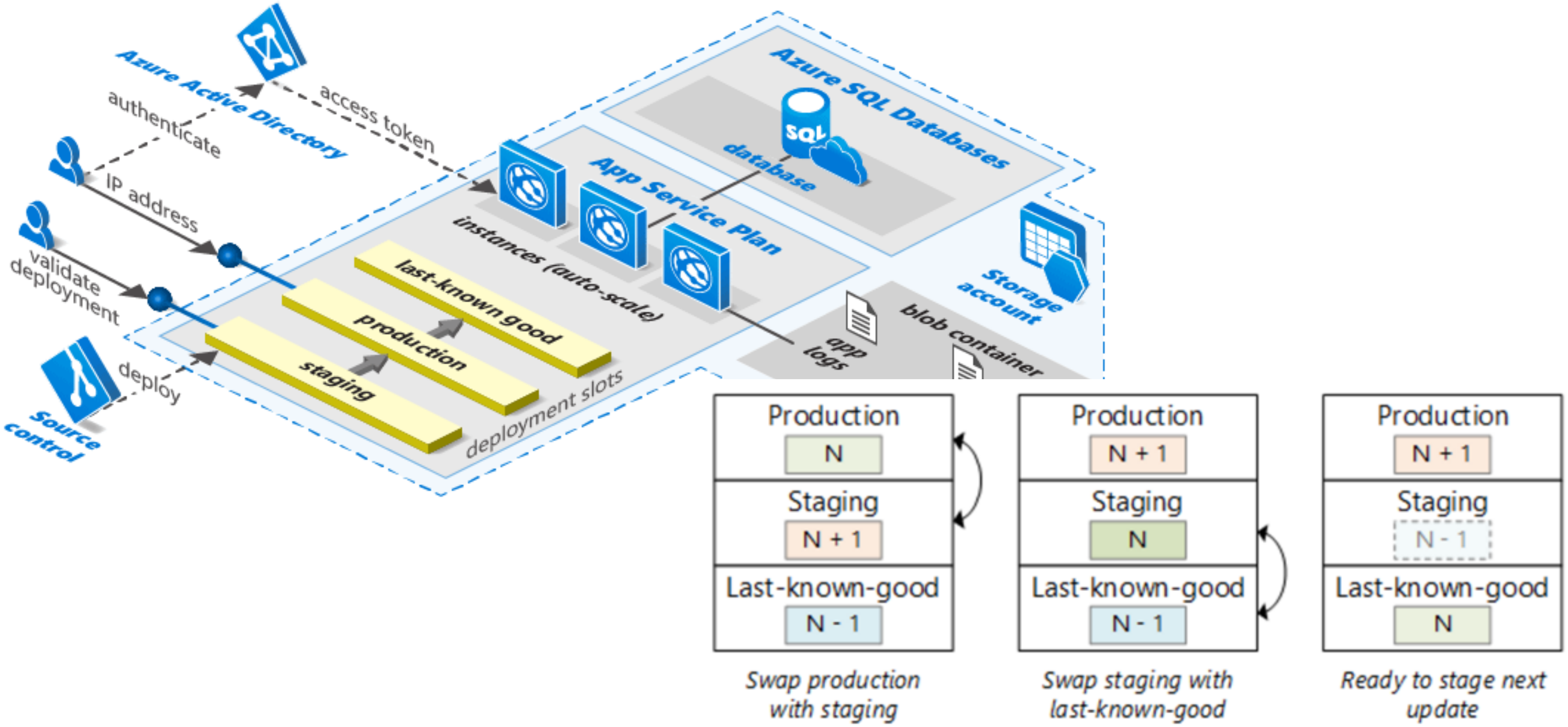
Blue/Green Deployment



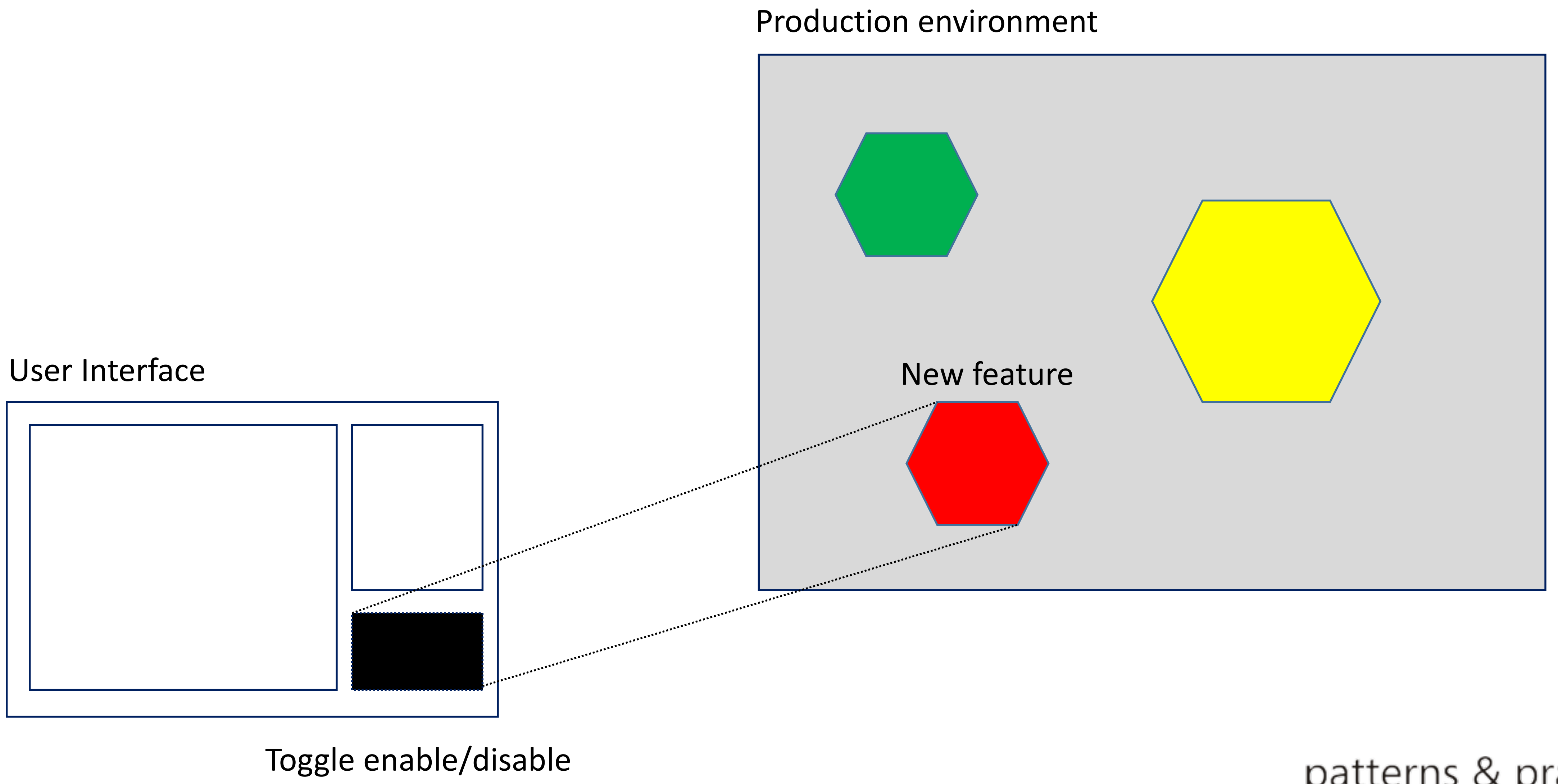
Canary release

patterns & practices
proven practices for predictable results

Deployment slots at App Service



Dark launching



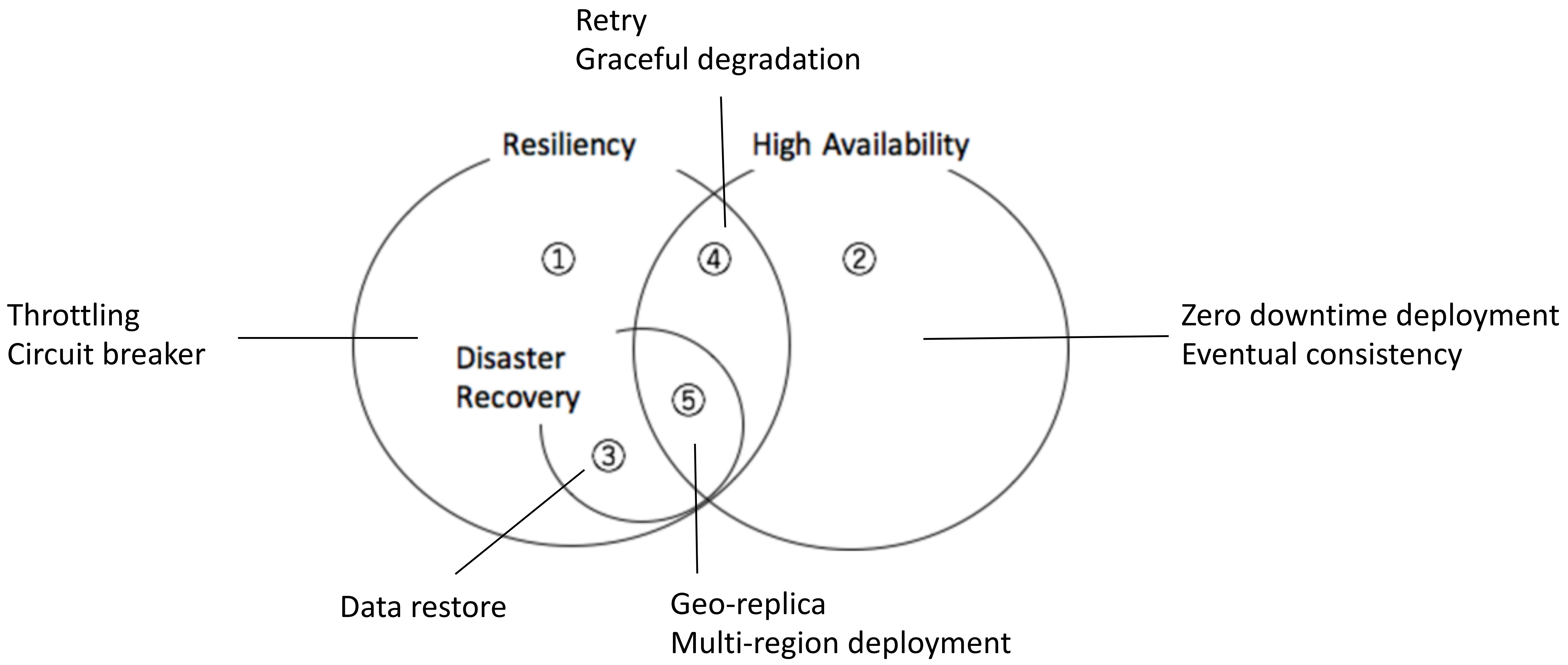
Resiliency checklist

- https://azure.microsoft.com/en-us/documentation/articles/guidance-resiliency-checklist/

Other resources

http://docs.microsoft.com/Azure

Resiliency / High Availability / Disaster Recovery


Retry
Graceful degradation
Resiliency
High Availability
①
④
②
Throttling
Circuit breaker
Zero downtime deployment
Eventual consistency
Disaster Recovery
⑤
③
Data restore
Geo-replica
Multi-region deployment