Microsoft Azure

Resiliência de Aplicações e Engenharia do Caos

George Luiz Bittencourt

Sobre mim

SCAN ME

George Luiz Bittencourt

Arquiteto de soluções cloud com mais de 20 anos de experiência em infraestrutura e desenvolvimento de software.

/glzbcrt

/glzbcrt

george.bittencourt@microsoft.com

Agenda

- Resiliência
- Responsabilidade Compartilhada
- Azure Well-Architected Framework
- Serviços
- Failure Mode Analysis
- SLA, SLO, SLI e Budget Error
- Técnicas/Boas Práticas
- Engenharia do Caos
- Azure Chaos Studio

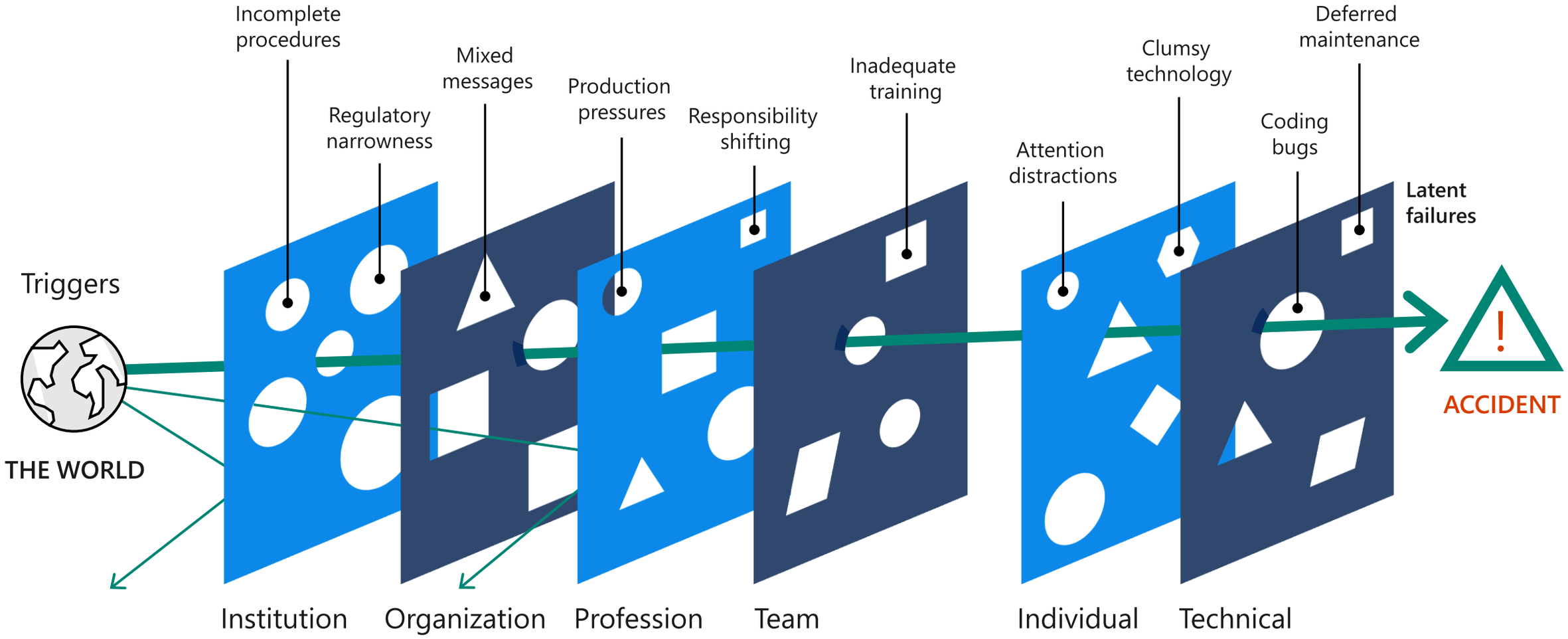
Resiliência

- Executar aplicações na nuvem é diferente de executar em ambientes *on-premisses*.
  - *No ambiente on-premisses focamos em adicionar o máximo de redundância possível.*
  - *Na nuvem, também adicionamos, mas aceitamos de que falhas vão acontecer.*
- **Reliability** é o grau de confiabilidade da aplicação.
  - *A percepção do usuário de uma aplicação que sempre está disponível e muito diferente de uma que cai com frequência.*
- **Resiliência** é como atingimos esse grau de confiabilidade utilizando técnicas, ferramentas, processos, etc.

O objetivo não é evitar falhas, mas sim responder as falhas de uma forma que evite indisponibilidade e a perda de dados.

Why do bad things happen?

Modified from
Reason, 1991



LAYERS OF DEFENSE

Azure Status

Responsabilidade Compartilhada

CLIENTE

- *Uso dos **serviços** com resiliência configurada.*
- *Backups e replicação.*
- *Atualizações de segurança em IaaS.*
- *Segurança Lógica.*

MICROSOFT

- *Disponibilização dos **serviços** com resiliência.*
- *Segurança Física.*
- *Atualizações de segurança em PaaS e SaaS.*

Customer need

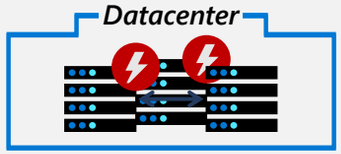
Improved availability

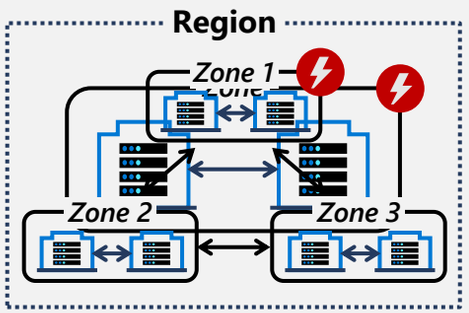
Build and run *highly-available* applications with near-zero RPO/RTO

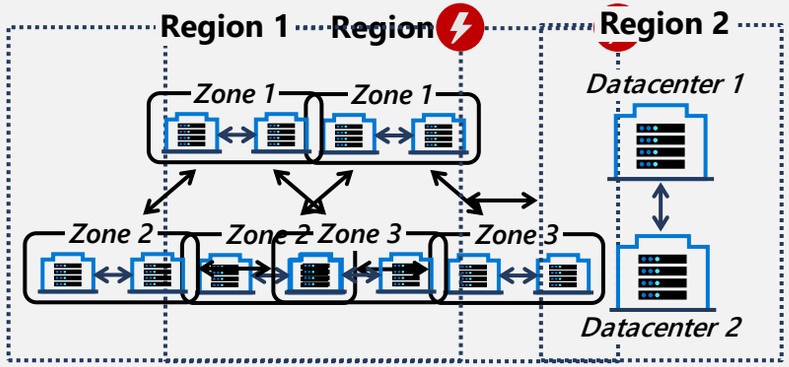
Implement *disaster recovery plans* with data residency and minimal RPO/RTO

Premium Storage	Availability Sets	Availability Zones	Azure Site Recovery / Region Pairs

Infrastructure (applications and data)







Scope of Impact...

SLA 99.9%	SLA 99.95%	SLA 99.99%	Industry-leading RPO/RTO
Isolated VM failure e.g. OS disk HDD issue	**Hardware failure** e.g. server rack issue	**Entire datacenter failure** e.g. power/network issue	**Entire region failure** e.g. natural disaster

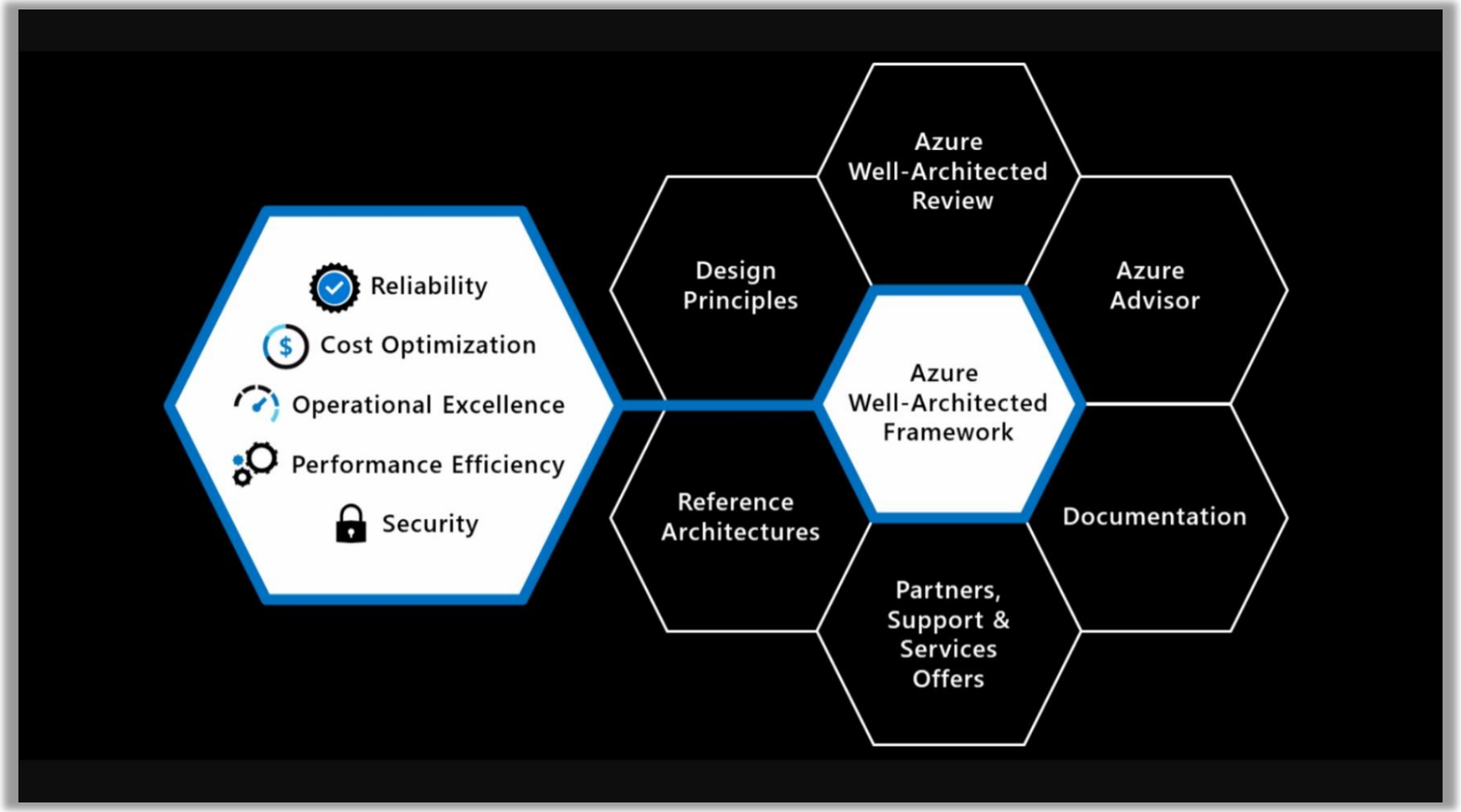
Data (stateful)

Accidental data loss | Data corruption | Ransomware | Rogue administrator

Backup

Go back to *restore* a healthy version of the data

Azure Well-Architected Framework



[Referência: Microsoft Azure Well-Architected Framework]

Azure Well-Architected Framework

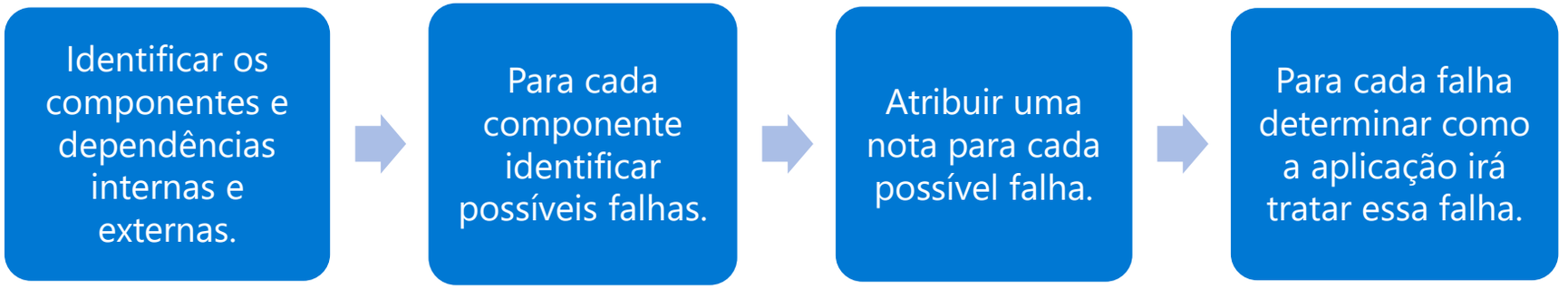
- **Design for Business Requirements**
  - *Os requerimentos de negócio é que determinam a confiabilidade de uma aplicação.*
  - *Quanto maior esse nível, mais caro fica.*
- **Design for Failure**
  - *Falhas são impossíveis de impedir. Abrace elas!*
  - *Se antecipe as possíveis falhas. Vamos ver como!*
- **Observe Application Health**
  - *Antes de mitigar falhas, precisamos monitorar o sistema.*
- **Drive Automation**
  - *A maior causa de indisponibilidade é erro humano, seja por falta de testes ou erros de configuração.*
  - *Automatizar retira o componente humano da equação.*
- **Design for self-healing**
  - *Capacidade do sistema de automaticamente se recuperar de falhas.*
  - *Requer uma maturidade maior com monitoramento e automação.*
- **Design for scale-out**
  - *Responder ao aumento de demanda adicionando novos recursos, ao invés de aumentar os recursos existentes.*
  - *Reduz o efeito das falhas, pois temos mais de uma unidade de processamento.*
  - *Com auto-scale temos economia financeira, pois temos somente o que precisamos.*

Serviços

- **Compute**
  - *Availability Zones*
  - *Availability Set*
  - *Kubernetes*
  - *Virtual Machine Scale Set*
- **Networking**
  - *Public IP*
  - *Azure Front Door*
  - *Azure Load Balancer*
  - *Azure Application Gateway*
  - *Azure Traffic Manager*
- **Storage**
  - *ZRS, GRS*
  - *Azure Site Recovery*
- **General**
  - *Region Pair*

DEMO

Failure Mode Analysis

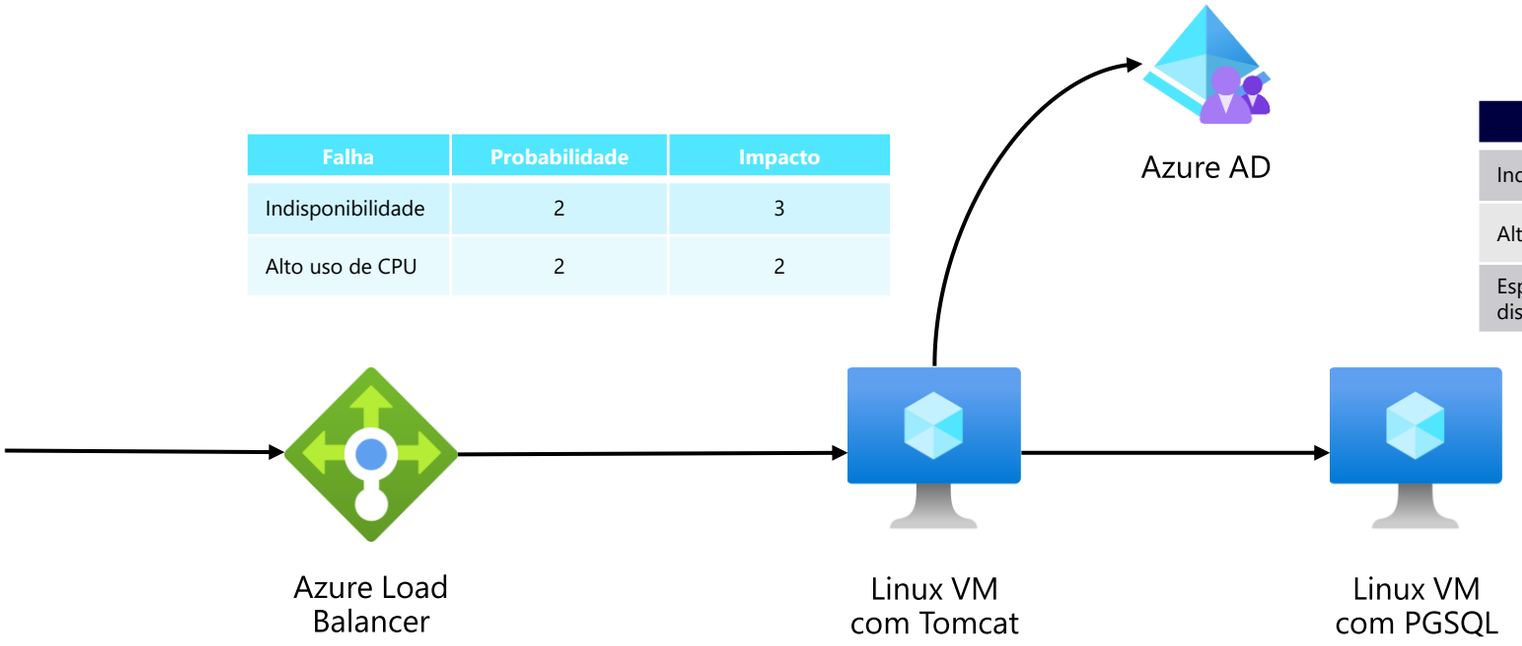


Referência: Failure mode analysis - Azure Architecture Center

Failure Mode Analysis

Falha	Probabilidade	Impacto
Indisponibilidade	1	3
Alto uso de CPU	2	3

Falha	Probabilidade	Impacto
Indisponibilidade	2	2
Alto uso de CPU	2	2

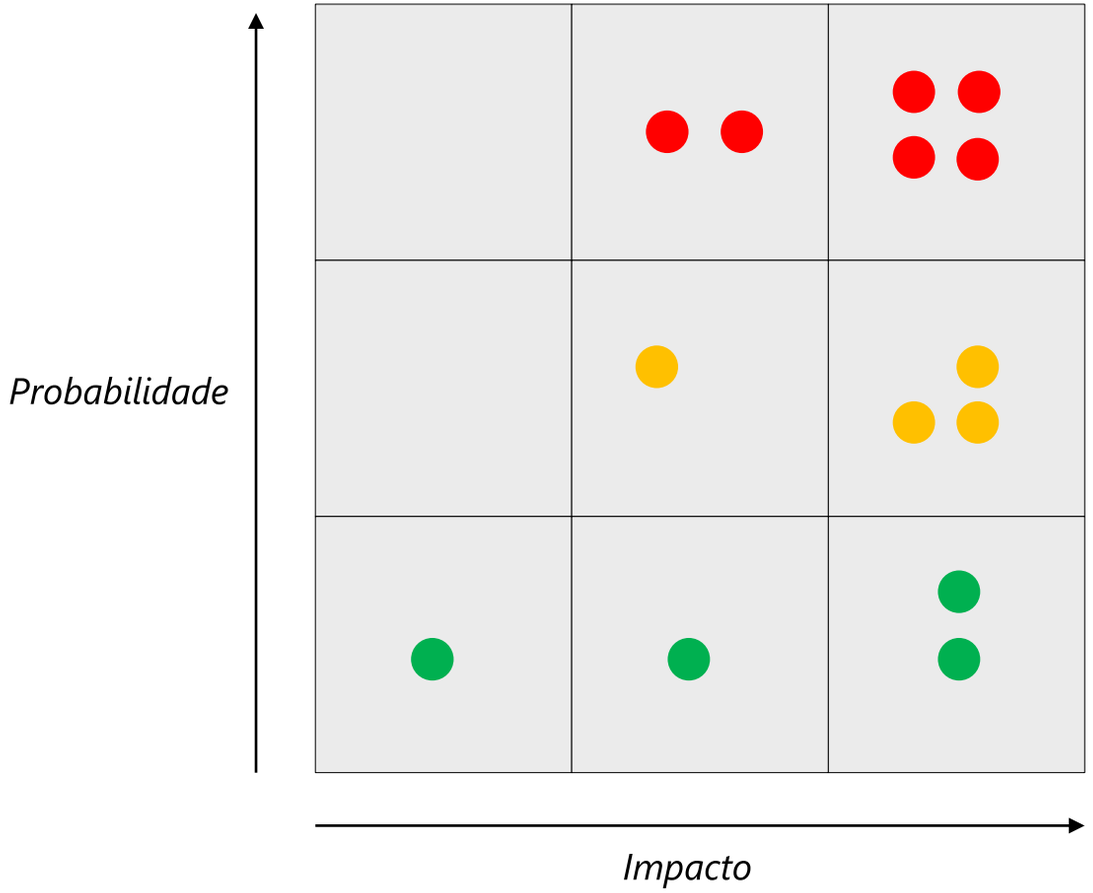


Falha	Probabilidade	Impacto
Indisponibilidade	2	3
Alto uso de CPU	2	3
Espaço livre em disco	1	3

Falha	Probabilidade	Impacto
Indisponibilidade	1	3
Erros manuais	2	3
Aumento de Latência	1	2

Falha	Probabilidade	Impacto
Indisponibilidade	2	3
Token expirado	1	3

Failure Mode Analysis



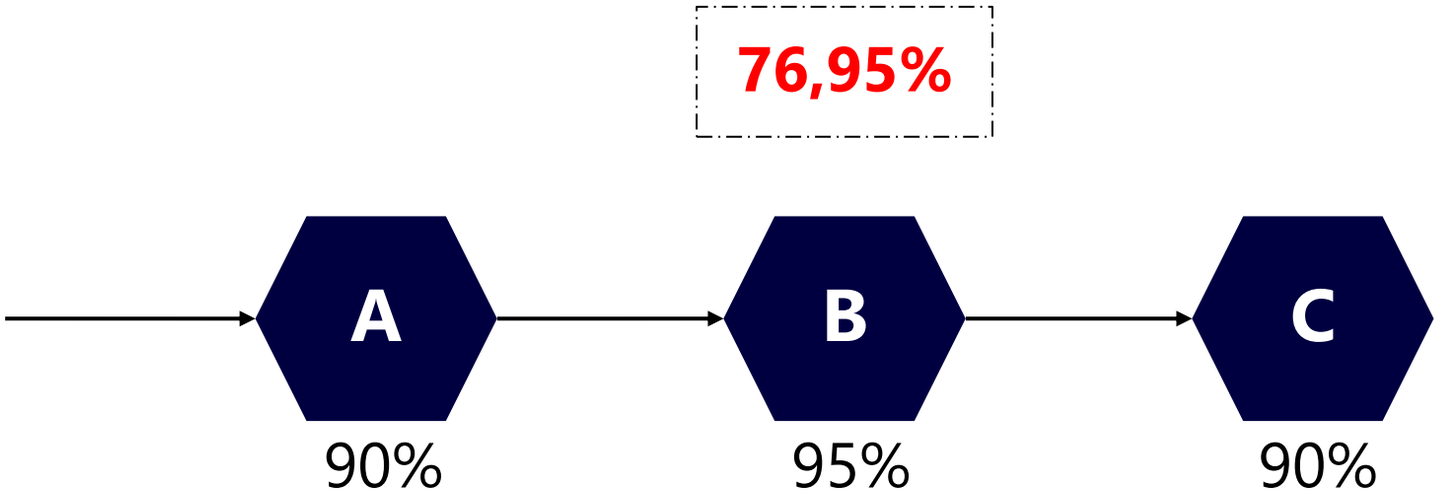
SLA, SLI, SLO e Error Budget

- SLA: define o acordo de nível de serviço entre as partes descrevendo os compromissos (SLIs e SLOs) e consequências de cada um.
- SLI: métrica monitorada dentro do SLA.
  - *Disk Latency*
  - *Requests per second*
  - *Network Availability*
- SLO: compromisso da parte com uma métrica.
  - *Disk Latency: 10 ms @ P95*
  - *Requests per second: 1 kRPS @ P90*
  - *Network Availability: 99,99%*
- Error Budget: é a parte do SLO que podemos utilizar para melhorias, manutenções, etc.
- Referência: Cloud monitoring service level objectives - Cloud Adoption Framework

SLA, SLI, SLO e Error Budget

% Disponibilidade	Error Budget
99,999	26s
99,99	4m 21s
99,95	21m 44s
99,9	43m 28s
99,5	3h 37m 21s
90	3d 26m 55s

SLA, SLI, SLO e Error Budget



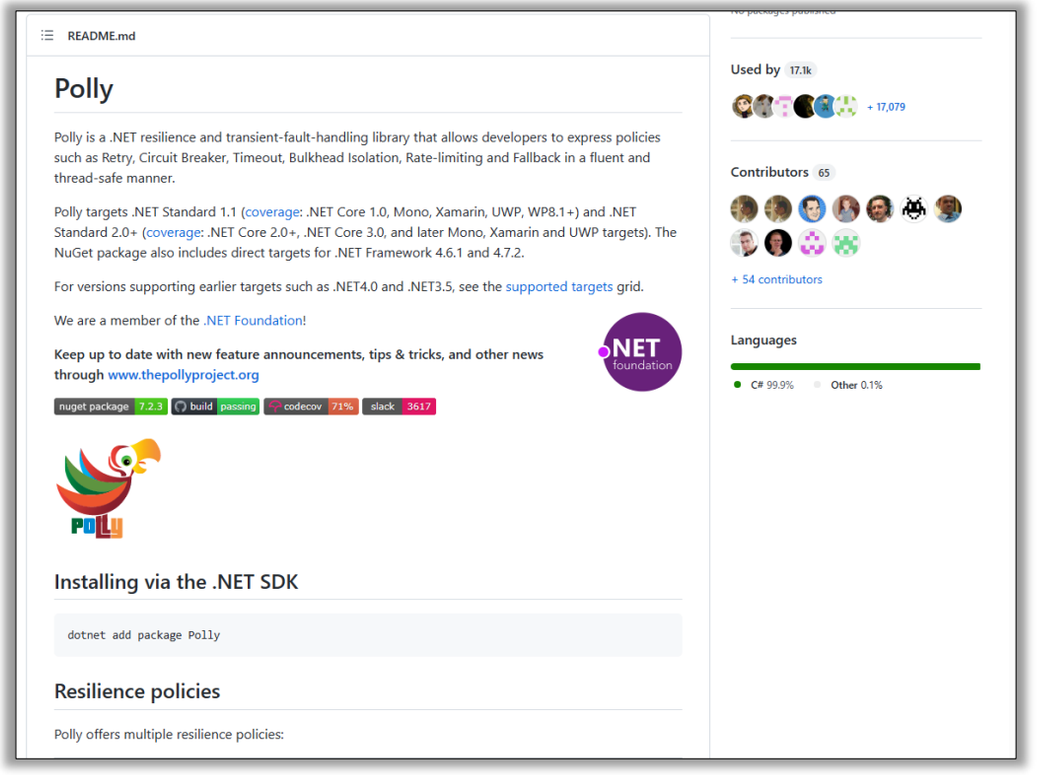
SLA Composto: 90 X 95 X 90 → 76,95%

Técnicas/Boas Práticas

- Timeout
  - *Configurar timeouts nas chamadas entre os componentes do sistema com valores adequados.*
  - *Geralmente os timeouts padrões são muito longos e acabamos usando recursos demais como threads, sockets, etc.*
- Retry
  - *Após o timeout expirar ou algum erro ser gerado, tentar novamente.*
  - *Adicionar um delay entre as chamadas para não ficarmos.*
- Circuit Breaker
  - *Se após várias chamadas não for possível acessar, abrir o circuito e não enviar mais requisições durante um período.*
  - *Evitamos colocar uma carga desnecessária no componente com problema até que o mesmo seja resolvido.*
- *Rate Limit*
  - *Limita a quantidade de chamadas por unidade de tempo. Exemplo: 10 chamadas por segundo.*
  - *Quando esse limite é atingido, ocorre throttling.*
- *Cache*
  - *Mantemos em uma memória rápida um dado custoso para gerar.*
  - *Se a fonte do dado estiver indisponível, podemos utilizar o cache para retornar o dado.*
- *Fallback*
  - *O que fazemos se falhar?*

Polly

- Biblioteca para .NET que implementa várias técnicas de resiliência, conhecidas como *cloud patterns*.
- Através de código utilizando uma interface fluente criamos uma *policy*.
- Podemos aproveitar uma mesma *policy* em vários pontos do sistema.
- Podemos combinar várias *policies*.

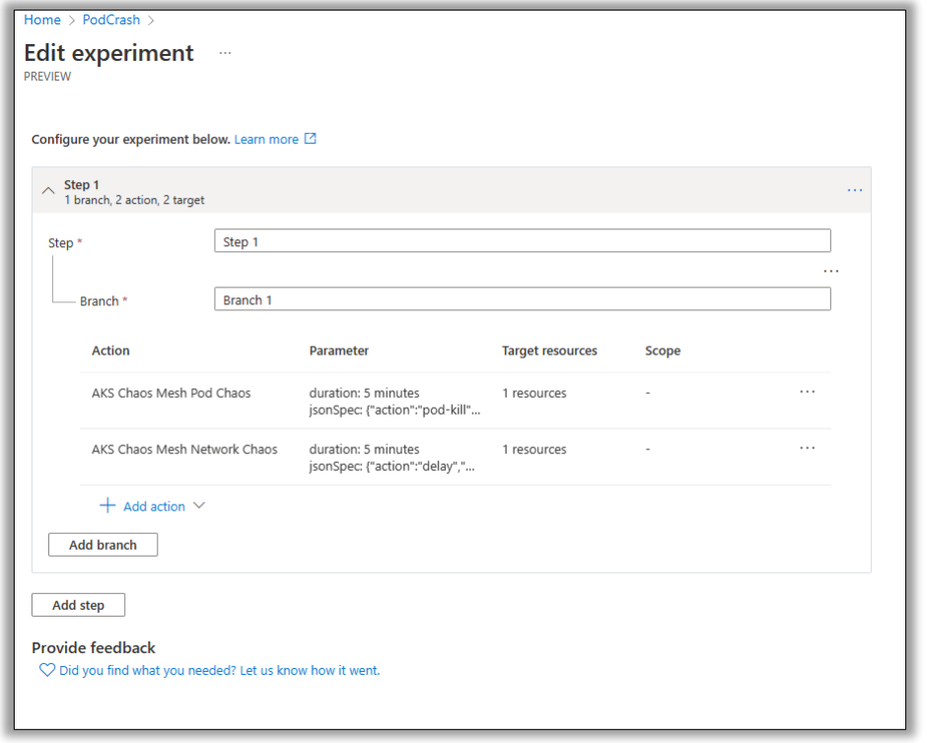


Engenharia do Caos

- Prática criada originalmente pelo Netflix.
- Tem por objetivo identificar fraquezas em sistemas antes que se tornem problemas críticos.
- Essas fraquezas são conhecidas como *dark debt*, que são as anomalias causadas por sistemas distribuídos complexos.
- Além da solução técnica outros componentes são validados como monitoramento, pessoas, guias, pessoas, etc.
- Pode ser utilizado tanto em *shift-left* quanto em *shift-right*.
- Identificamos hipóteses através de testes de *what-if* e desenvolvemos experimentos para validar.
- Alguns ferramentas:
  - Netflix/chaosmonkey
  - Chaos Mesh para Kubernetes
  - Azure Chaos Studio
  - Chaos Toolkit

Azure Chaos Studio

- Produto em preview.
- Tem dois componentes, os *experimentos* e os *targets*.
- Antes de executar um experimento os *targets* precisam ser habilitados.
- Alguns experimentos exigem que um agente seja instalado no destino previamente.
- Um experimento tem *steps* e *branchs* e *actions*.
- *Steps* são executados em sequência e *branchs* em paralelo.
- A *action* executa alguma das *faults* existentes em um grande catálogo.
- Cada *action* tem sua própria parametrização.
- A execução pode ser agendada pelo Logic Apps.



DEMO

Microsoft Azure

EOF

/glzbcrt

/glzbcrt

george.bittencourt@microsoft.com

SCAN ME