Code Island
CLOUD

Maximizando a Resiliência na nuvem com Chaos Engineering

Thiago Finardi
Platform Engineer - SRE
Chaos Engineer

Code Island CLOUD

Agradecimento:

Patrocínio:

evoluum

CDB Data Solutions

Dati Soluções em TI

unisul

Apoio:

COFFOPS

Thiago Finardi

ADS / Security / IoT Sistemas Distribuídos

Platform Engineer - SRE

Komodo - Chaos Engineering & Performance Tests

Thiago Finardi
@tfinardi
www.finardi.me

Agenda:

- Desafios com sistemas distribuidos
- O que é Chaos Engineering
- Tipos de ataques
- Introdução ao AWS Fault Injection Simulator (FIS)
- Funcionalidades
- Casos de uso
- Workshop

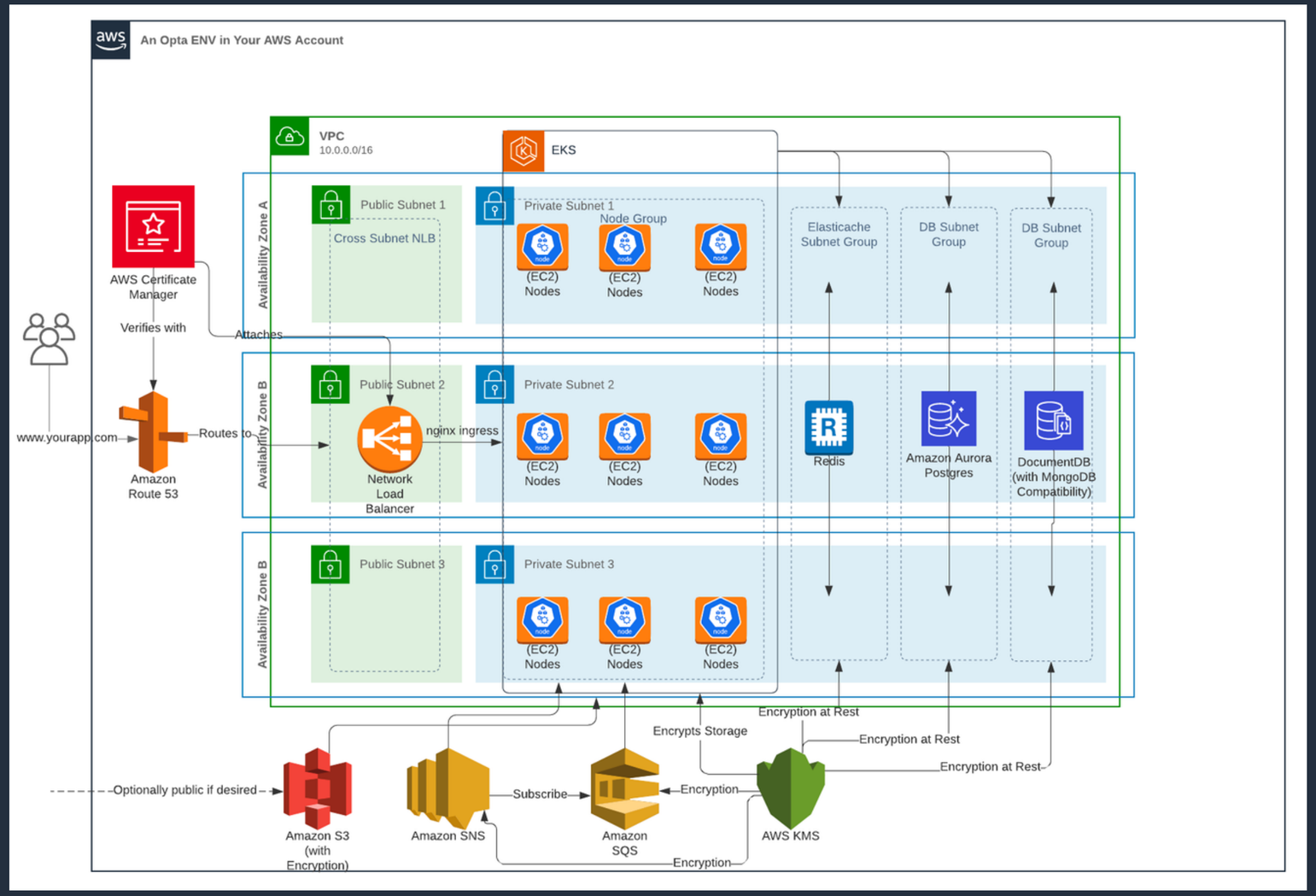
Code Island CLOUD

Desafios com sistemas distribuídos

Code Island CLOUD

Sistemas distribuídos são complexos

Como garantir que todas as peças e contratos estão em conformidade?



https://aws.amazon.com/builders-library/challenges-with-distributed-systems/

Code Island CLOUD

Testes tradicionais não são suficientes

Teste de unidade de componentes

Testado isoladamente para garantir que a função atende às expectativas

Teste funcional de integrações

Cada path é testado para garantir que os resultados obtidos são os esperados

Teste end to end

A jornada é testada para garantir que os resultados obtidos são os esperados

TESTES = VERIFICAÇÃO DE UMA CONDIÇÃO CONHECIDA

Code Island CLOUD

Testes com Engenharia do Caos

Testes com foco em garantir a resiliência de sistemas distribuídos em ambientes complexos.

Code Island CLOUD

O que não é chaos engineering?

O que é Engenharia do Caos?

Chaos Engineering é uma abordagem sistemática e disciplinada para a experimentação em sistemas distribuídos, com o objetivo de identificar e solucionar problemas antes que eles se tornem interrupções inesperadas para o usuário final.

"O processo de realizar experimentos controlados em um sistema para aprender como ele se comporta em condições de falha, de forma a identificar e corrigir problemas antes que eles afetem o usuário final."

(Fonte: "Chaos Engineering: System Resiliency in Practice" de Casey Rosenthal e Nora Jones, publicado em 2020)

Code Island CLOUD

Chaos engineering

Melhorar a resiliência e o desempenho
Maior confiança na capacidade do sistema

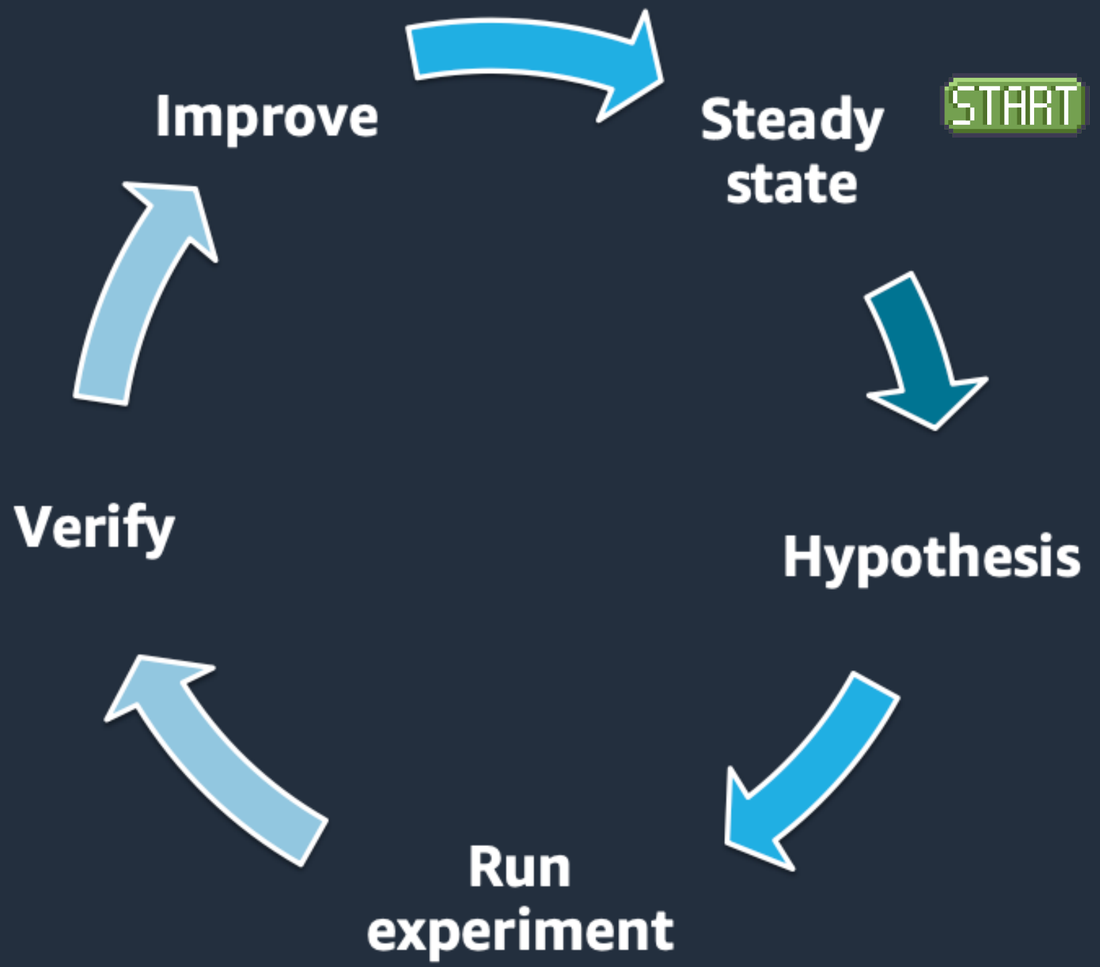
Descubrir problemas ocultos
Melhoria da experiência do usuário final

Expor pontos cegos
Monitoramento, observabilidade e alarmes

Processo contínuo de melhoria e aprendizado

Code Island CLOUD

Fases da engenharia do caos



Alguns tipos de ataques

	ATTACK	TECHNICAL INITATIVES	BUSINESS INITATIVES
RESOURCE	CPU	• Stress testing • Validate autoscaling • Validate monitoring and alerting • Validate resource quotas • Right-size infrastructure	• Prepare for peak traffic events • Streamline cloud migrations • Cost optimization
	Memory	• Prepare for memory leaks and memory intensive workloads • Right-size infrastructure • Validate autoscaling • Prepare for a cloud migration	• Prepare for peak traffic events • Streamline cloud migrations • Cost optimization
	Disk	• Right-size infrastructure • Test automatic processes and provisioning • Validate monitoring	• Cost optimization
	I/O	• Right-size infrastructure • Test disk-heavy workloads • Validate caches	• Prepare for peak traffic events • Improve UX • Streamline product launches

Ebook: How to use Gremlin's Attacks in a mature chaos engineering and reliability practice

Code Island CLOUD

Alguns tipos de ataques

	ATTACK	TECHNICAL INITATIVES	BUSINESS INITATIVES
STATE	**Process killer**	• Validate redundancy and failover • Validate cluster integrity	• Maximize uptime
	Shutdown	• Validate replication and failover • Validate cluster availability and integrity • Validate automatic restarts	• Maximize uptime
	Time travel	• Prepare for time-based events like DST, leap years, and “end of epch” problems • Test TLS certificate expiration • Test clock sync between systems	• Security and compliance

Ebook: How to use Gremlin’s Attacks in a mature chaos engineering and reliability practice

Code Island CLOUD

Alguns tipos de ataques

	ATTACK	TECHNICAL INITATIVES	BUSINESS INITATIVES
NETWORK	Blackhole	• Test dependency failures • Validate cluster availability and load balancing • Validate monitoring and alerting • Test redundancy and failover	• Prepare for peak traffic • Maximize uptime • Business continuity and disaster recovery planning • Streamline cloud migrations
	DNS	• Prepare for DNS outages	• Maximize uptime
	Latency	• Streamline cloud migrations • Optimize network performance • Validate load balancing • Validate timeout and retry thresholds • Test concurrency control	• Prepare for peak traffic events • Improve UX • Streamline cloud migrations
	Packet Loss	• Test data throughput and integrity • Test graceful degradation • Validate quality of service (QoS)	• Prepare for peak traffic events • Meet data integrity and compliance requirements • Improve UX

Ebook: How to use Gremlin's Attacks in a mature chaos engineering and reliability practice

Code Island CLOUD

Por que a engenharia do caos é difícil?

1

Juntar diferentes ferramentas e scripts caseiros

2

Agentes ou bibliotecas necessários para começar

3

Difícil garantir a segurança

4

Difícil de reproduzir eventos do “mundo real” (múltiplas falhas ao mesmo tempo)

Pré requisito:

OBSERVABILIDADE

É fundamental ter uma observabilidade dos componentes para que a equipe possa avaliar os resultados dos testes de caos e tomar ações corretivas caso necessário.

A falta de observabilidade irá dificultar a detecção de problemas e tornar a engenharia do caos menos efetiva.

Code Island CLOUD

AWS Fault Injection Simulator

Code Island CLOUD

AWS Fault Injection Simulator

Serviço de engenharia do caos totalmente gerenciado

Fácil de começar a usar

Condições de falha do mundo real

Barreiras de proteção

Code Island CLOUD

Fácil de começar a usar

- **Não há necessidade de integrar várias ferramentas e scripts caseiros ou instalar agentes**
- **Use o Console de gerenciamento da AWS ou a AWS CLI**
- **Use modelos de experiência pré-existentes e comece em minutos**
- **Compartilhe facilmente com outras pessoas**

Code Island CLOUD

Condições de falha do mundo real

Execute experimentos em sequência de eventos ou em paralelo

Alvo todos os níveis do sistema (host, infraestrutura, rede, etc.)

Falhas reais injetadas no nível do painel de controle do serviço!

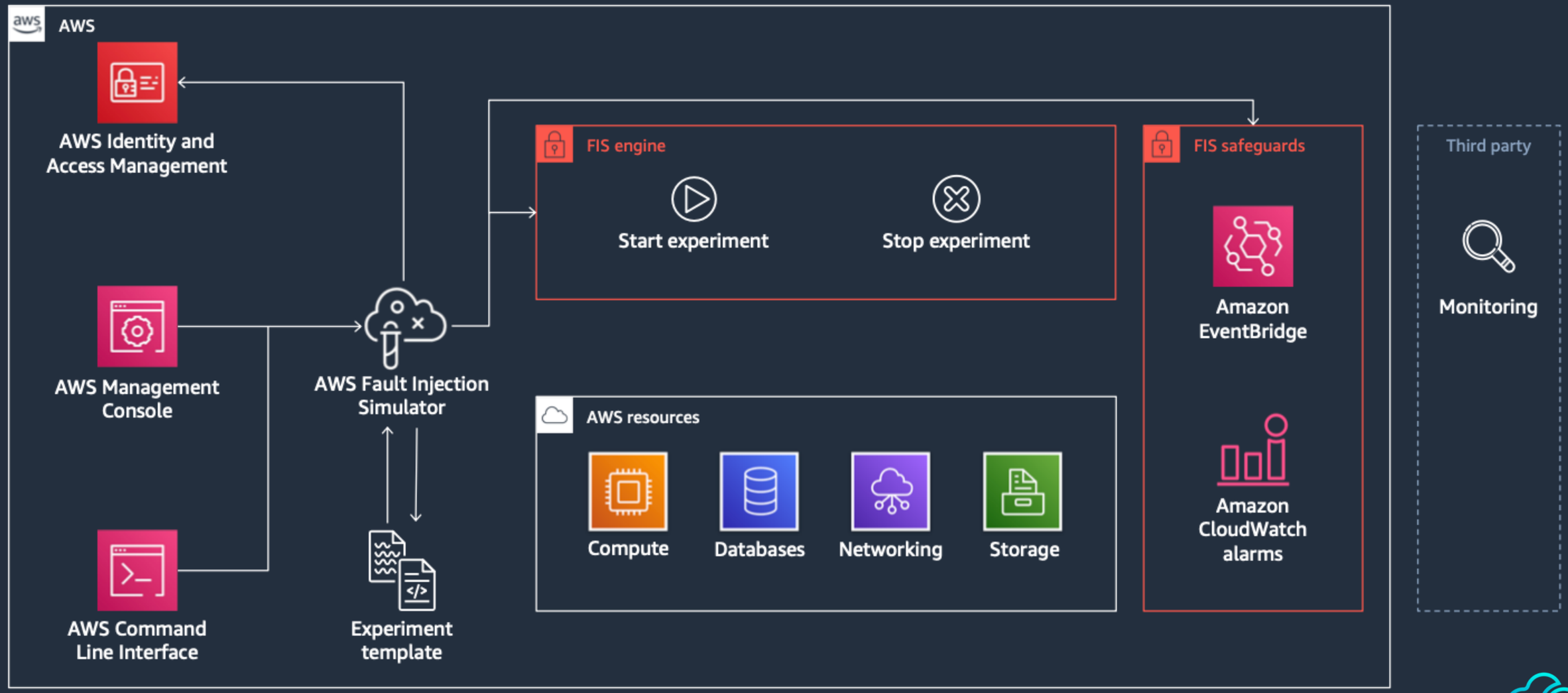
Code Island CLOUD

Barreiras de proteção

- Alarmes de “condições de parada”
- Integração com Amazon CloudWatch
- Rollbacks integrados
- Permissões IAM refinadas

Code Island CLOUD

AWS Fault Injection Simulator



Componentes

Ações

Alvos

Templates de experimentos

Experimentos

Code Island CLOUD

Ações

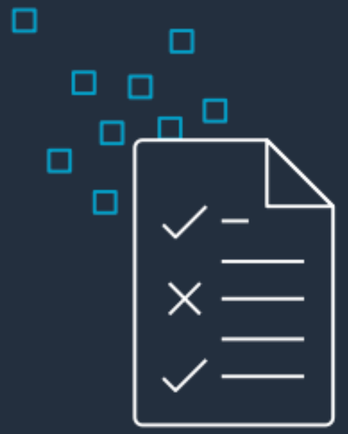
Ações são as ações de injeção de falhas executadas durante um experimento

aws:<service-name>:<action-type>

As ações incluem:

- Tipo de falha
- Recursos direcionados
- Tempo relativo a quaisquer outras ações
- Parâmetros específicos de falha, como duração, comportamento de reversão ou limites

Code Island CLOUD



Ações

```
"actions": {
        "StopInstances": {
                "actionId": "aws:ec2:stop-instances",
                "parameters": {
                    "startInstancesAfterDuration": "PT2M"
                },
                "targets": {
                        "Instances": "RandomInstancesInAZ"
                }
        },
        "Wait": {
                "actionId": " aws:fis:wait",
                "parameters": {
                    "duration": "PT1M",
                },
                "startAfter": [
                        "StopInstances"
                ]
        },

}
```

Code Island CLOUD

Alvos

Os **alvos** definem um ou mais recursos da AWS nos quais se pretende realizar uma ação

Os alvos incluem:

- Tipo do recurso
- IDs, tags e filtros dos recursos
- Modo de seleção (Ex: ALL, RANDOM)

Code Island CLOUD

Alvos

```
"targets": {
        "RandomInstancesInAZ": {
                "resourceType": "aws:ec2:instance",
                "resourceTags": {
                        "Env": "test"
                },
                "filters" : [
                        {
                                "path": "Placement.AvailabilityZone",
                                "values": ["us.east.1a"]
                        },
                        {
                                "path": "State.Name",
                                "values": ["running"]
                        },
                        {
                            "path": "VpcId",
                            "values": ["vpc-0123456789"]
                        }
                ]
                "selectionMode": "COUNT(2)"
        }
```

Code Island CLOUD

Templates de experimento

Os **templates de experimento** definem um experimento e são usados na request de início do experimento

Os templates de experimento incluem:

- Ações
- Alvos
- Condições de alarmes de parada
- IAM role
- Descrição
- Tags

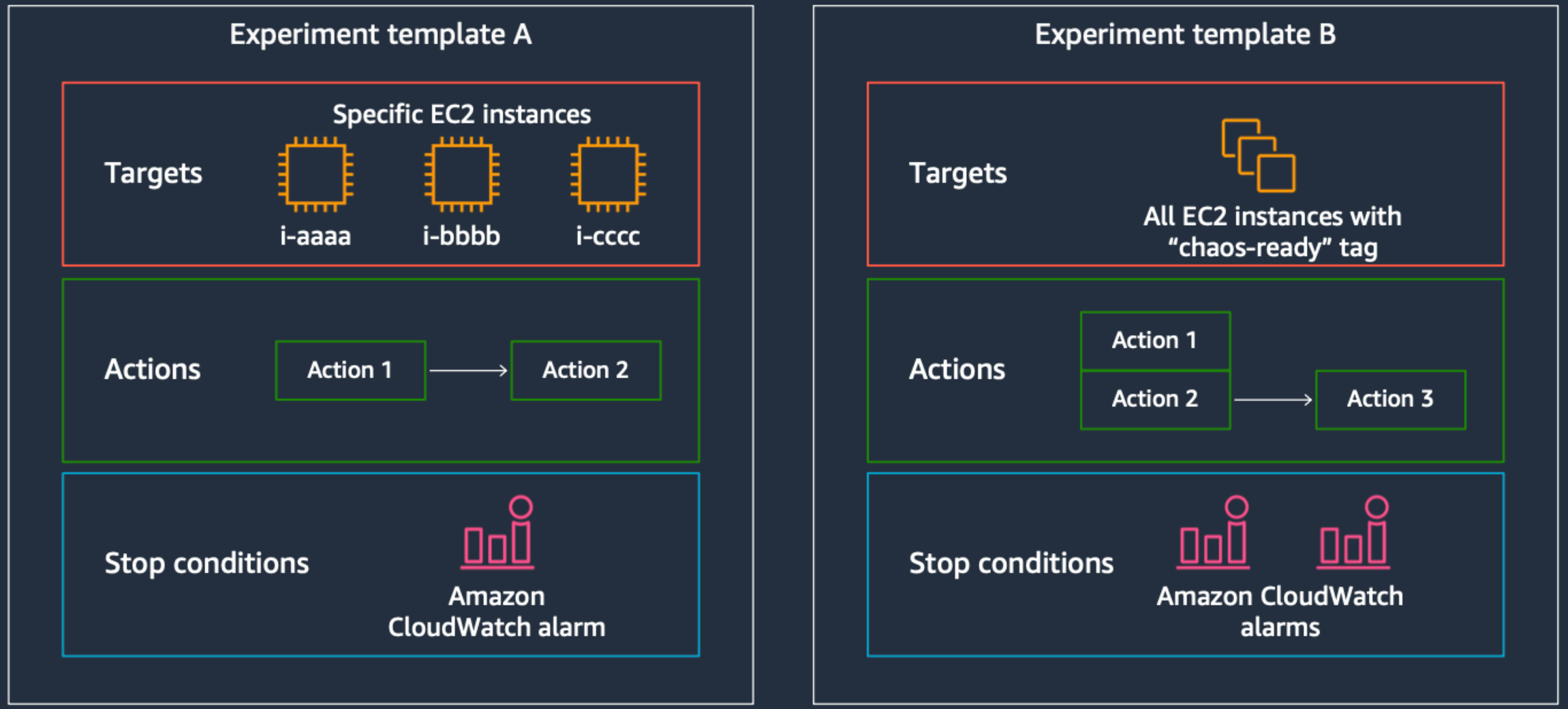
Code Island CLOUD

Templates de experimento

- Name
- Description
- IAM role
- Stop conditions
- Targets
- Actions

```
{
    "tags": {
         "Name": "StopAndRestartRandomInstance"
    },
    "description": "Stop and Restart One Random Instance",
    "roleArn": "arn:aws:iam::0123456789:role/MyFISExperimentRole",
    "stopConditions": [
         {
             "source": "aws:cloudwatch:alarm",
             "value": " "arn:aws:cloudwatch:us-east-1:0123456789:alarm:No_Traffic"
         }
    ],
    "targets": {
         "myInstance": {
             "resourceTags": {
                 "Env": "test"
             },
             "resourceType": "aws:ec2:instance",
             "selectionMode": "COUNT(1)"
         }
    },
    "actions": {
         "StopInstances": {
             "actionId": "aws:ec2:stop-instances",
             "description": "stop the instances",
             "parameters": {
                 "startInstancesAtEnd": "true",
                 "duration": "PT2M",
             },
             "targets": {
                 "Instances": "myInstance"
             }
         }
     }
}
```

Templates de experimento



Code Island CLOUD

Experiments

Os **experimentos** são snapshots do modelo de experimento quando ele foi lançado pela primeira vez com algumas adições

Os experimentos incluem:

- Snapshot do experimento
- Hora de criação e início
- Status
- ID de execução
- ID do template de experimento
- IAM role ARN

Code Island CLOUD

Injeções de falhas suportadas

- Server error (EC2)
- Stop, reboot, and terminate instance(s) (EC2)
- API throttling
- Increased memory or CPU load (EC2)
- Kill process (EC2)
- Latency injection (EC2)
- Container instance termination (ECS)
- Increase memory or CPU consumption per task (ECS)
- Terminate nodes (EKS)
- Database stop, reboot, and failover (RDS)
- Network disruption, EBS pause, e outros...

Code Island CLOUD

Casos de uso

Experimentos únicos

Game days periódicos

Experimentos automátizados

Code Island CLOUD

Experimentos automatizados

Experimentos agendados

Experimentos acionados por eventos

Experimentos de entrega contínua

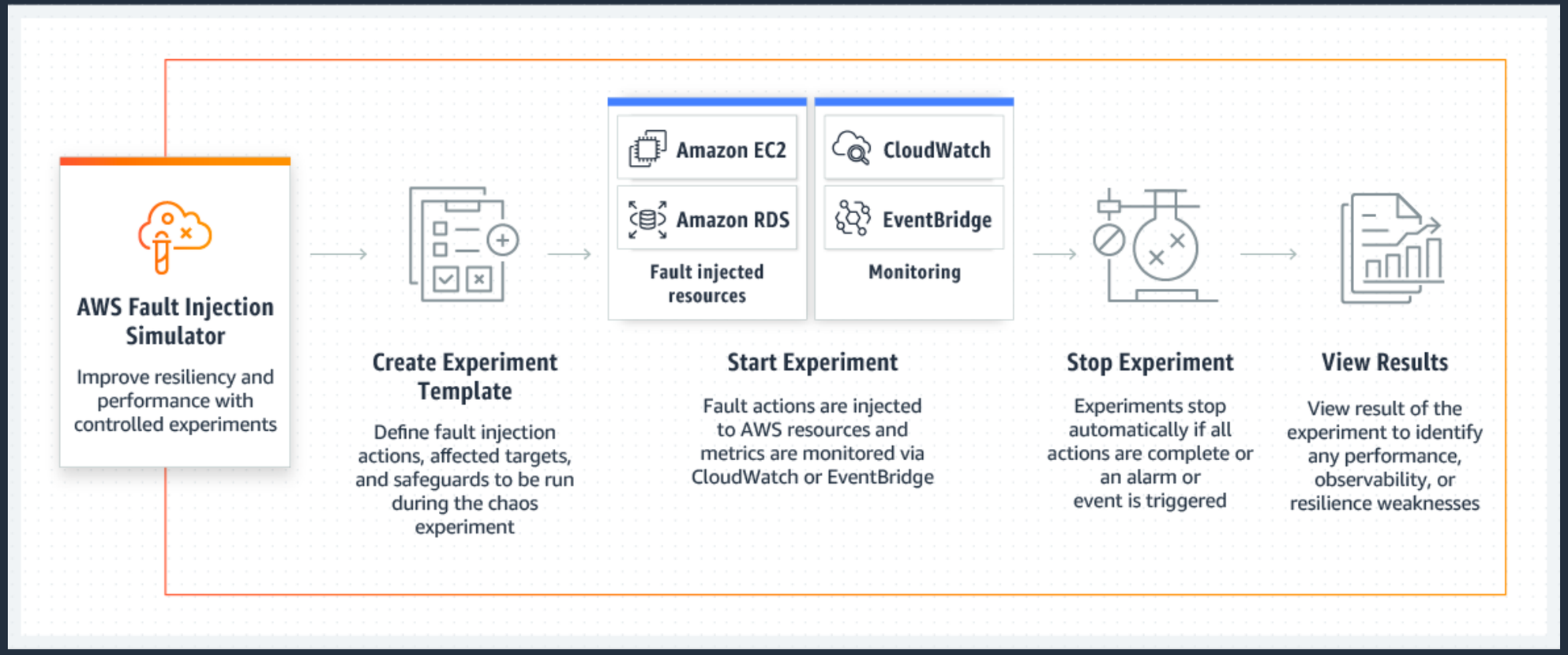
Code Island CLOUD

Workshop

Workshop Studio

Discover and participate in AWS workshops and GameDays

workshops.aws



https://chaos-engineering.workshop.aws/

Code Island CLOUD

Outras soluções

Gremlin

harness

Chaos Engineering

Litmus

steadybit

ChaosToolkit

Chaos Monkey

Chaos Mesh

Azure Chaos Studio

Istio

ChaosBlade

Code Island CLOUD

Links

AWS Well-Architected Framework
https://aws.amazon.com/architecture/well-architected/

AWS Fault Injection Simulator
https://aws.amazon.com/fis/

AWS Chaos Engineering Workshop
https://chaos-engineering.workshop.aws/

Documentação do AWS FIS
https://docs.aws.amazon.com/fis/

AWS FIS Exemplos
https://github.com/aws-samples/aws-fault-injection-simulator-samples

Code Island CLOUD

Dúvidas?

Obrigado!

Thiago Finardi
@tfinardi