AWS Fault Injection Service で AZ 障害を体験しよう

可用性構成の検証

Mari Tsukamoto

Amazon Web Services Japan G.K.
Solution Architect

塚本　真理(Tsukamoto Mari)

アマゾンウェブサービスジャパン
技術統括本部 ソリューションアーキテクト

趣味:

- 息子と電車めぐり、ゲーム

好きな AWS サービス:

- Amazon RDS / Aurora 、 AWS Amplify

構築した可用性構成、障害のときも本当に使える？

机上の設計だけで満足していませんか？

障害が起きたときのオペレーションは試していますか？

Chaos Engineering の 原則

PRINCIPLES OF CHAOS ENGINEERING

Last Update: 2018 May

Chaos Engineering is the discipline of experimenting on a system
in order to build confidence in the system's capability
to withstand turbulent conditions in production.

“カオスエンジニアリングとは
「対象とするシステムが本番環境における不安定な状況を耐えることができる」
という自信を構築するために
当該システムにおいて実施する 実験 の規律です。”

https://principlesofchaos.org (英語), https://principlesofchaos.org/ja/ (日本語)

Chaos Engineering の前に考えるべきこと

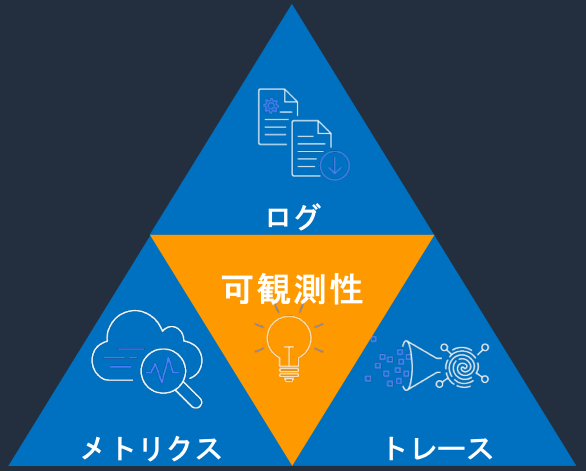
2つの観点が重要
→ ①現在の改善点を確認 & ②可観測性の確保

①現在のシステムの改善点を確認する

AWS Well-Architected Framework

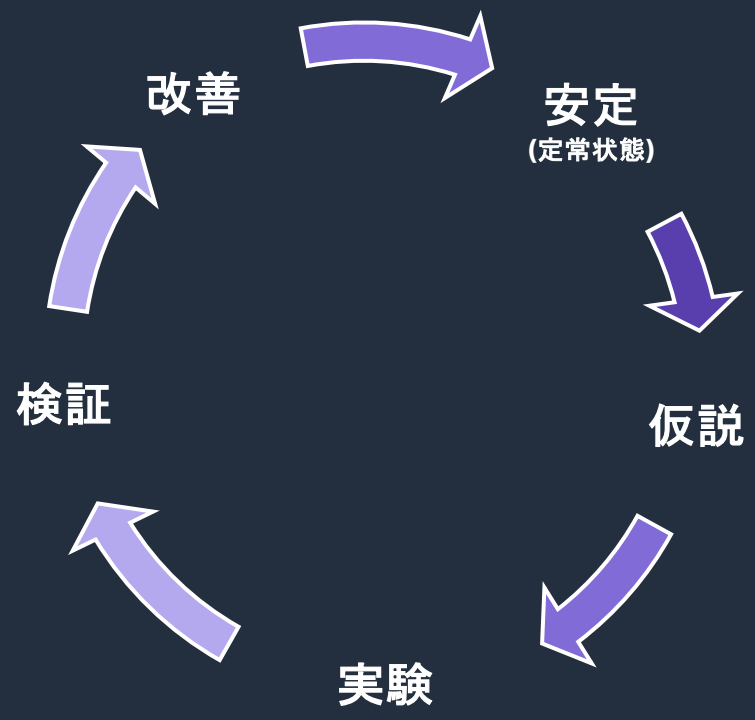
https://aws.amazon.com/well-architected/

②可観測性(Observability)を確保する



Chaos Engineering：要点

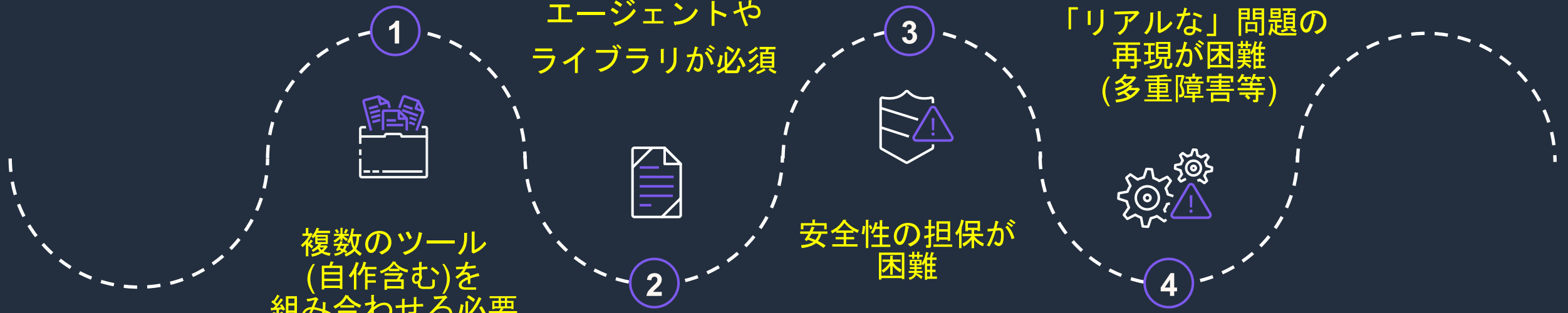
実際に Chaos Engineering を 取り入れるためには、様々な障壁がある



1. 定常状態を定義する
2. 実験が始まっても定常状態が継続する仮説を立てる
   "Xを実行したとしても、このシステムの定常状態に変化はないはずである"
   (「こうなるはず しかし 確証がない」
   → Chaos Engineeringにおける「実験」検証ポイント)
3. 現実世界を反映する変数(Chaos)をシミュレートし実験する
   (→ サーバーダウン、レイテンシ増大 などの事象を注入)
4. 実験結果を検証し、必要なら改善する
   (→ 記録と仮説の反証 → 反証されたら学びを改善に活かす。反証されなかったらその仮説が「自信」となる)

Chaos Engineering に伴う困難さ

実際に Chaos Engineering を 取り入れるためには、様々な障壁がある



Chaos Engineeringは「実験」を制御する ツール の活用が不可欠

AWS Fault Injection Service

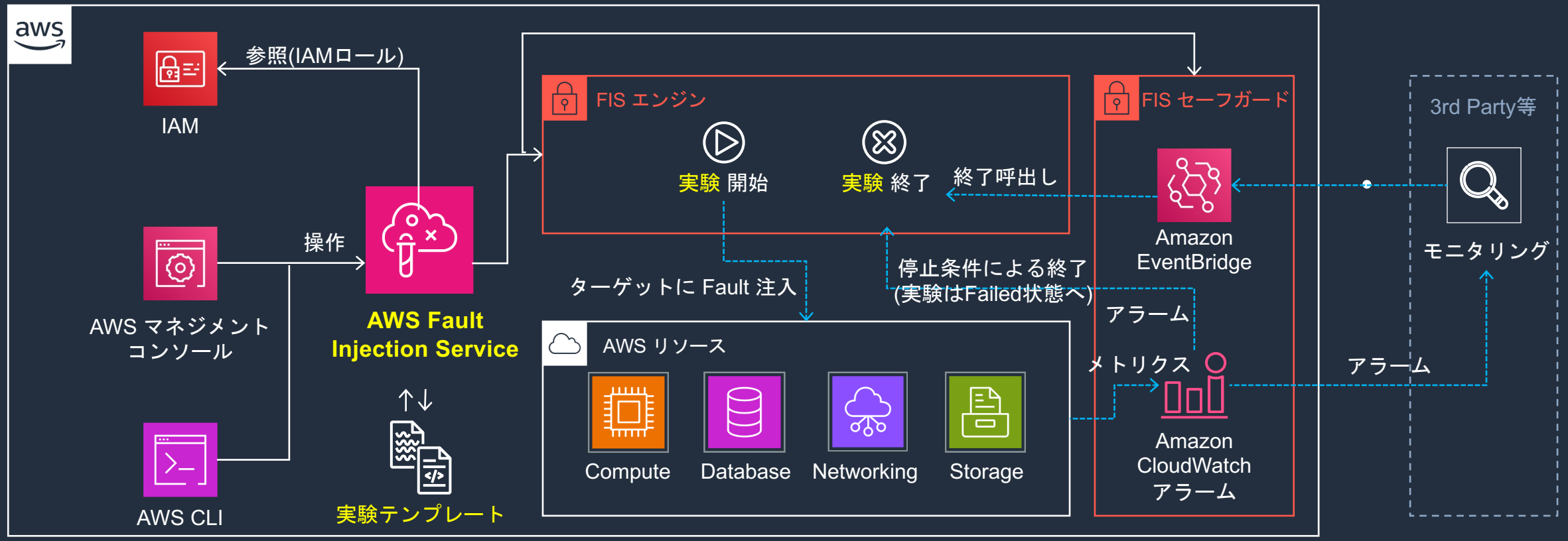
AWS Fault Injection Service (FIS) - 位置づけ

AWS Resilience Hub の1つの機能として、
レジリエンス実現のための「実験を管理するサービス」として位置づけられる

※「AWS Fault Injection Simulator」から2023年11月に名称変更(略称はFISで不変)

FIS - 概観

リソース横断での「実験(Experiment)」開始による 障害状況 の注入と
指定条件をトリガーとした 終了 をフルマネージドサービスとして提供



FIS - 実験テンプレート

最初に「実験テンプレート(Experiment Template)」を定義
→ 実験のシナリオを表現 (実験テンプレートを元に「実験」を開始)

実験に用いるIAMロール(必要な権限を許可)

この実験のアクションが実行される対象のAWSリソース(IDまたはタグで選択。その上で数や割合、タグの場合はリソース属性値でフィルタ可)
(タグ指定を推奨)

実験アクションを最大20指定(種類は次ページ参照)
各アクションは登録ターゲットを1つ参照
「次のあと開始」の項目指定で順序実行も可能

開始した実験を自動で強制終了させる条件を指定
(指定を推奨)

※AWS管理コンソールUI以外に JSON/YAML形式で定義可能(CLI)

FIS - 利用可能な実験アクションタイプ(1/4)

実験テンプレート内に「アクション」を複数追加して内容を設定
→ 以下のアクションが利用可能 (適宜機能追加される)

#	AWSサービス	アクション	FISアクションタイプ名
1	EC2	インスタンスのStop, Reboot, Terminate	aws:ec2:stop-instances, reboot-instances, terminate-instances
2		インスタンスのCPU負荷の上昇	aws:ssm:send-command/AWSFIS-Run-CPU-Stress
3		インスタンスのルートボリューム枯渇	aws:ssm:send-command/AWSFIS-Run-Disk-Fill
4		インスタンスのI/O負荷の上昇	aws:ssm:send-command/AWSFIS-Run-IO-Stress
5		インスタンスのメモリ負荷の上昇	aws:ssm:send-command/AWSFIS-Run-Memory-Stress
6		インスタンスのプロセスkill	aws:ssm:send-command/AWSFIS-Run-Kill-Process
7		インスタンスの通信ポート上にネットワーク通信におけるブラックホールの注入	aws:ssm:send-command/AWSFIS-Run-Network-Blackhole-Port
8		インスタンスにネットワーク遅延の注入	aws:ssm:send-command/AWSFIS-Run-Network-Latency
9		特定ソースへのネットワーク遅延の注入	aws:ssm:send-command/AWSFIS-Run-Network-Latency-Sources
10		インスタンスにパケットロス事象の注入	aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss
11		特定ソースへのパケットロス事象の注入	aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss-Sources

※SSMコマンド(aws:ssm:send-command系)の実行は対象が SSMマネージドノード として構成されている必要あり
※一部SSMコマンドは実行時に必要な追加パッケージ(コマンド)をダウンロード&インストールする
※#2～#5のアクションはLinuxプラットフォームにのみ対応・依存コンポーネント導入が前提

FIS - 利用可能な実験アクションタイプ(2/4)

実験テンプレート内に「アクション」を複数追加して内容を設定
→ 以下のアクションが利用可能 (適宜機能追加される)

#	AWSサービス	アクション	FISアクションタイプ名
12		スポットインスタンスの中断通知	aws:ec2:ec2:send-stop-instance-interruptions
13	EC2	APIで不十分なキャパシティエラーの発生	aws:ec2:api-insufficient-instance-capacity-error
14		ASGで不十分なキャパシティエラーの発生	aws:ec2:asg-insufficient-instance-capacity-error
15	CloudWatch	アラームの状態設定	aws:cloudwatch:assert-alarm-state
16	DynamoDB	暗号化グローバルテーブルの複製を停止	aws:dynamodb:encrypted-global-table-pause-replication
17	EBS	EBSのI/O停止	aws:ebs:pause-volume-io
18	ECS	コンテナインスタンスのドレイン	aws:ecs:drain-container-instances (ECSのUpdateContainerInstancesState APIを実行)
19		タスクの停止	aws:ecs:stop-task
20		タスクのCPU負荷の上昇	aws:ecs:task-cpu-stress
21		タスクのプロセスkill	aws:ecs:task-kill-process
22		タスクの通信ポート上にネットワーク通信におけるブラックホールの注入	aws:ecs:task-network-blackhole-port
23		タスクにネットワーク遅延の注入	aws:ecs:task-network-latency

FIS - 利用可能な実験アクションタイプ(3/4)

実験テンプレート内に「アクション」を複数追加して内容を設定
→ 以下のアクションが利用可能 (適宜機能追加される)

#	AWSサービス	アクション	FISアクションタイプ名
24		タスクにパケットロス事象の注入	aws:ecs:task-network-packet-loss
23	EKS	ChaosMesh または Litmusの実験を実行	aws:eks:inject-kubernetes-custom-resource
24		PodのCPU負荷の上昇	aws:eks:pod-cpu-stress
25		Podの削除	aws:eks:pod-delete
26		PodのI/O負荷の上昇	aws:eks:pod-io-stress
27		Podのメモリ負荷の上昇	aws:eks:pod-memory-stress
28		Podの通信ポート上のネットワーク通信におけるブラックホールの注入	aws:eks:pod-network-blackhole-port
29		Podにネットワーク遅延の注入	aws:eks:pod-network-latency
30		Podにパケットロス事象の注入	aws:eks:pod-network-packet-loss
31		EC2ノードグループのWorkerノード終了	aws:eks:terminate-nodegroup-instances (指定割合のノードを停止)
32	ElastiCache	Redisレプリケーショングループノード終了	aws:elasticache:interrupt-cluster-az-power

FIS - 利用可能な実験アクションタイプ(4/4)

実験テンプレート内に「アクション」を複数追加して内容を設定
→ 以下のアクションが利用可能 (適宜機能追加される)

#	AWSサービス	アクション	FISアクションタイプ名
33	RDS	データベースのReboot, Failover	aws:rds:reboot-db-instances, failover-db-cluster
34	Systems Manager	指定インスタンスにSSM Document の実行	aws:ssm:send-command
35		指定Automation Documentの実行	aws:ssm:start-automation-execution
36	S3	S3バケットレプリケーションの停止	aws:s3:bucket-pause-replication
37	IAM	AWS APIスロットリング, エラー, 利用不可	aws:fis:inject-api-throttle-error, aws:fis:inject-api-internal-error, aws:fis:inject-api-unavailable-error ※現時点での対象APIは EC2 API のみ
38	Networking	ネットワーク接続性の混乱	aws:network:disrupt-connectivity, ※サブネットに対してNACLを用いたN/W接続拒否
39		クロスリージョン接続性の混乱 (Public IP/VPCピアリング)	aws:network:route-table-disrupt-cross-region-connectivity
40		クロスリージョン接続性混乱 (Transit Gatewayピアリング)	aws:network:transit-gateway-disrupt-cross-region-connectivity
41	その他	待機 (複数アクション間の時間調整用)	aws:fis:wait

FIS - シナリオライブラリ & シナリオ

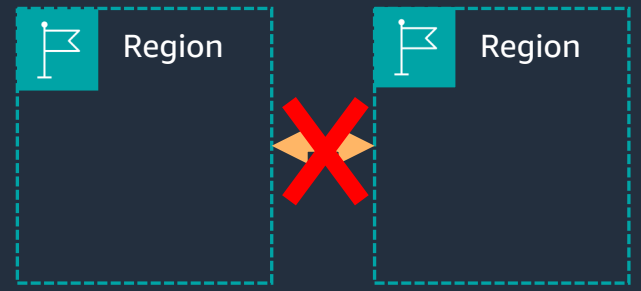
複数アクションの組み合わせによる一連の処理シナリオを提供

#	カテゴリー	シナリオ名	説明
1	EC2 stress	Instance failure	• EC2インスタンスの停止により、インスタンスダウン事象を再現
2		Disk	• ディスク使用率上昇
2		CPU	• CPU使用率上昇
3		Memory	• メモリ使用率上昇
4		Network Latency	• ネットワークレイテンシー上昇
5	EKS stress	Pod Delete	• 1つ以上のPodを削除
6		Disk	• Podのディスク使用率上昇
7		CPU	• PodのCPU使用率上昇
8		Memory	• Podのメモリ使用率上昇
9		Network Latency	• Podのネットワークレイテンシー上昇
10	AZ Availability	Power Interruption	• AZ電源障害の疑似的再現(インスタンスダウン、新規EC2インスタンス起動不可、サブネット間ネットワーク通信の混乱、EBSボリュームI/Oの停止等の発生)
11	Cross-Region	Connectivity	• リージョン間通信不可事象の擬似的再現

一般利用開始

AWS Fault Injection Serviceで2つのイベントシナリオを公開

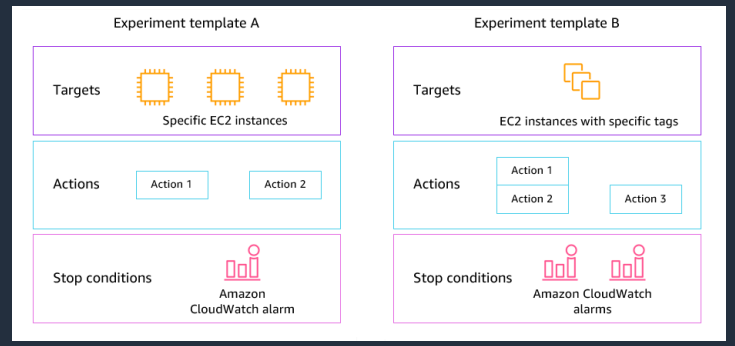
- AWS Fault Injection Service(FIS)で多くの要望を頂いていた2つイベントシナリオを公開
- The AZ Availability: Power Interruption
  - ひとつのAZで完全に電源を喪失した場合を想定。マルチAZのアプリケーションの振る舞いを試験できる
- Cross-Region: Connectivity
  - リージョン間の接続が切断され他リージョンのリソースにアクセスできない場合を想定。マルチリージョンアプリケーションの振る舞いを試験できる
- FISが利用可能な全てのリージョンにて

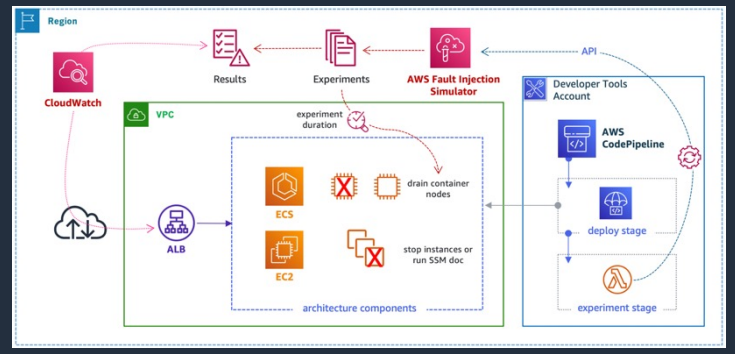


一般利用開始

AWS Fault Injection Serviceがマルチアカウントに対応

- 複数のAWSアカウントに跨るアプリケーションに対して障害シナリオを実行可能に
  - オーケストレーターアカウントからマルチアカウント用Experimentテンプレートを用いて容易に実行可能
  - リソースタグで対象を指定
  - IAMロールで詳細な権限管理が可能
- AWS Fault Injection Serviceをサポートする全てのリージョンで利用可能





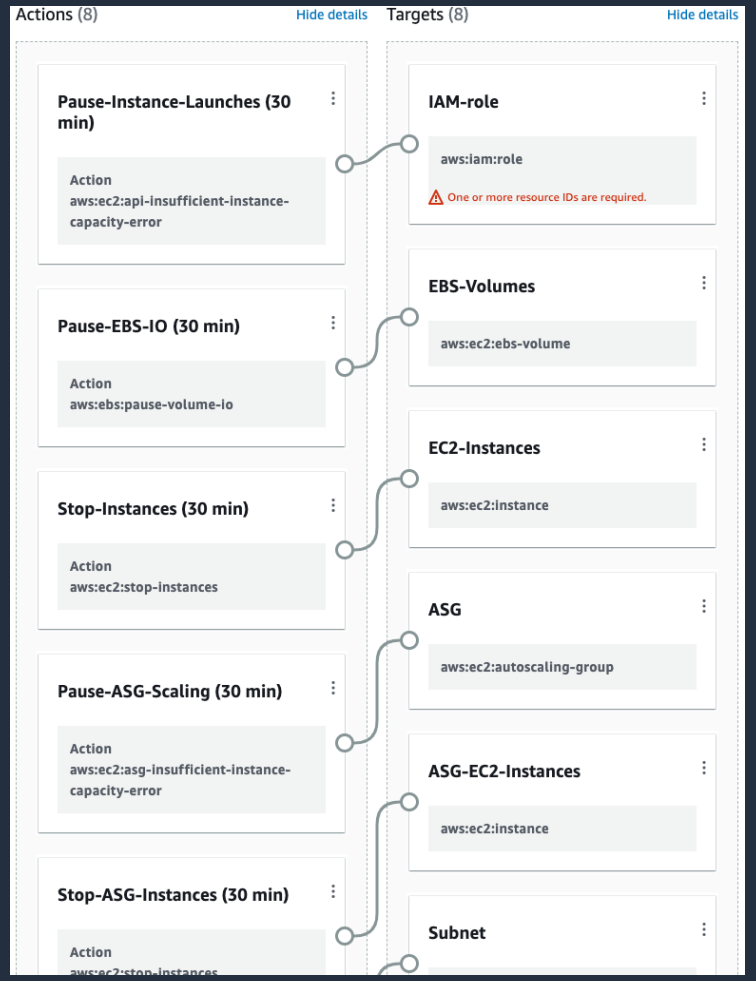
※Fault Injection Simulatorから名前が変わりました

デモ
AZ電源障害の疑似的再現

The AZ Availability: Power Interruption

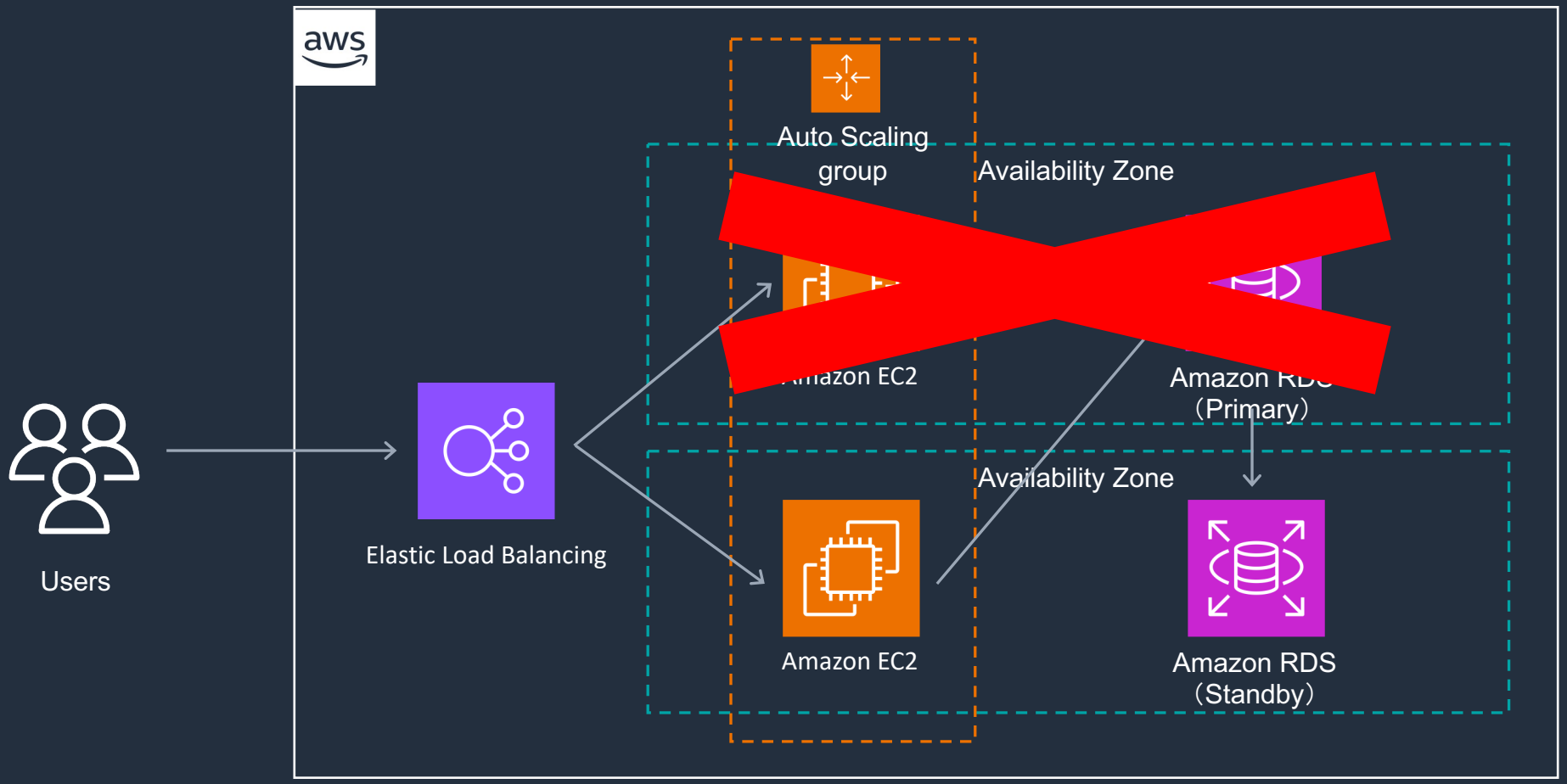
- 以下 8つのアクションを実施
  - Stop-Instances
  - Stop-ASG-Instances
  - Pause Instance Launches
  - Pause ASG Scaling
  - Pause Network Connectivity
  - Failover RDS
  - Pause ElastiCache Redis
  - Pause EBS I/O

※Fargateは未サポート、ElactiCacheもRedisのみ



https://docs.aws.amazon.com/en_us/fis/latest/userguide/az-availability-scenario.html

マルチAZ構成のアプリケーションでのAZ障害



まとめ

まとめ

FIS は システムの振る舞いを継続的に実験し、
現実の信頼性、可用性への自信を獲得するツールとしてご利用可能

- AWS re:invent 2023でのアップデートで、AZ障害やクロスリージョンでの接続障害の再現が可能に

- 推奨アプローチ
  - 見つけられる課題 (参考: AWS Well-Architected フレームワーク)の解決と、可観測性の確保からまずは取り組む
  - 「定常状態」と仮説を定義し、制御された実験を行う
  - 小さく初めて広げていく
    - 開発環境・ステージング環境での実施
    - 社内イベントとして始め、それを自動化していく
    - いつでも停止可能な仕組みを整備する

Thank you!