Well-Architected-Framework

**アーキテクチャ設計基礎**

**AZの選択**

１リージョンに対して２つが基本（３つは逆効果：高スト的な問題）

複数AZにロードバランサーを使ってEC2インスタンス配置

S3を使ってバックアップを取る

**VPC**

２つ以上のVPCを作ることが基本となる

　 　　例：開発環境用VPCと本番環境

システム単位や組織単位で分ける

分割方法：マルチVPC方式　マルチアカウント方式

サブネットでの分割

CIDR範囲で分割

パブリックサブネット

インターネットと接続が必要なリソースを揃える

インターネットとのアクセス制御に利用する

プライベートサブネット

インターネットから隔離することでセキュリティを高める

分割の基準はWebアクセスの要否

サブネットサイズは/24以上が推奨されている

１つのAZに対してパブリック・プライベートサブネットを１つずつ配置基本

VPC間接続の設計

オンプレとの接続ポイントをどうするのか

Well-Architcted-Framework

試験範囲

レジリエントアーキテクチャの設計

Reliability

高パフォーマンスアーキテクチャの設計

Performance Efficiency

セキュアなアプリケーションとアーキテクチャの設計

Security

コスト最適化アーキテクチャの設計

Cost Optimization

オペレーショナルエクセレンスを備えたアーキテクチャの定義

Operational Excellence

↓

５つの設計原則

信頼性・パフォーマンス効率・安全性・コスト最適化・運用上の優秀性

Well-Architcted-Framework

①Well-Architcted-Frameworkホワイトペーパー（試験範囲）

②AWSのSAまたは認定パートナーによる支援制度

③セルフチェック向けのWell-Architcted-Framework

ベストプラクティスを利用することで最適解や改善点を理解する

要件定義

最適な要件検討に利用する

設計

適切な設計方式

構築

リスクや改善点をホワイトペーパで確認していく

運用

適用中に不定期にホワイトペーパを確認してレビューを実施する

常にホワイトペーパを見ながら確認していくことがAWSから求められている

Reliability（信頼性）

インフラストラクチャーサービスの障害復旧の自動化など軽減設計

復旧手順のテストによる検証

需要変化に応じた水平方向へのスケーラビリティに高可用性の確保

キャパシティの推測をやめる

モニタリングと自動化を進める

信頼性の基本対応領域

基盤・変更管理・障害管理

基盤：

IAM 権限管理

VPC 複数のVPCやサブネットを作成する

AutoScaling

ELB ヘルスチェックやスケーラビリティの自動化

Cloud Foemation 依存関係などのセットアップ自動化する

変更管理：

Cloud Trail

AWS Config 変更点など更新処理をモニタリング

障害管理：

Cloud Watch 常にモニタリングしていく

Performance Efficiency：パフォーマンス効率

システム要件を満たすためのコンピューティングリソースの効率化

システム要件やAWSサービスの進化に応じてAWSインフラの効率化を推進

―先端技術の一般化（AIなどの活用）

―グローバル化を即座に達成（各国リージョンを使う）

―サーバレスアーキテクチャの利用（Lambdaが推奨されている）

―より頻繁な実験

パフォーマンス効率の基本対応領域

コンピューティング・ストレージ・データベース・容量と時間のトレードオフ

コンピューティング：

Auto Scaling・Lambda

ストレージ：

EBS・S3・Glacier・EFS

データベース：

RDS・DynamoDB・Elastic serch・Aurora・Redshift

容量と時間のトレードオフ

Cloud Front・Elastic Cache

Security：セキュリティ

全てのレイヤーでのセキュリティを適用

アクセス追跡・モニタリングを確実な実施

条件ドリブンのアラートをトリガーにしセキュリティイベントへの応答を自動化

AWS責任共有モデルに基づく対象範囲の保護に集中する

セキュリティのベストプラクティスの自動化

　・ソフトウェアベースのセキュリティ設定を使用する

迅速でコスト効率の良いスケーリングを安全に実行する

　・仮想サーバのカスタムベースラインイメージによる新サーバへの適用自動化

　・インフラストラクチャー全体のテンプレ化による管理

パフォーマンス効率の基本対応領域

データ保護・権限管理・インフラ保護・検出制御

データ保護：

ELB・EBS・S3・RDS・KMS

暗号化やキーマネージドサービスの活用

権限管理：

IAM・MFA

インフラ保護：

VPCでのインフラ保護

検出制御：

Cloud Trail・Cloud Watch・AWS GuardDuty・Amazon Inspector

Cost Optimization：コスト最適化

不必要なリソース軽減

透明性のある費用賦課

マネージド型サービスの利用によるコスト軽減

固定の償却コストを変動コストへと転換

スケールによるコストメリット

データセンターへの投資不要化

Cost Optimizationの基本対応領域

需要と供給の一致

コスト効率の高いリソース

支出の認識

継続した最適化

需要と供給の一致：

AutoScaling

コスト効率の高いリソース：

EC2購入方式(リザーブドなど)・TrustestAdvisor

支出の認識：

Cloud Watch・見積ツール

継続した最適化：

AWS最新情報の取得・TrustestAdvisor

Operational Excellence：運用上の優秀性

コードに基づく運用実施

ビジネス目的に沿った運用手順

定期的かつ小規模で増加的な変更実施

予期せぬイベントへの応答テスト

運用イベントと障害からの学習

運用手順を最新のものに保持すること

Operational Excellenceの基本対応領域

準備・運用・進化

準備：

CloudFormation・Codeシリーズ・RunbookPlaybook

自動化ツール　　　　　　　　　ガイドブック

運用：

各種モニタリングツール

進化：

常に新しいサービスの活用を考えてベストプラクティスに準拠する

AWSベストプラクティス

１１の原則

・スケーラビリティの確保

需要の変化に対応できるアーキテクチャを設計する

・環境の自動化

システムの安定性・整合性及び組織の効率性を改善する主要サービス活用

・使い捨てリソースの使用

サーバなどのコンポーネントを一時的なリソースとして利用

・コンポーネントの疎結合

相互依存性を減らした構成にして障害児の影響を減らす

・サーバではなくサーバレス

サーバレスのアーキテクチャによる効率的な設計や運用

・最適なDB選択

ワークロードに応じたデータベース技術を利用する

・増大するデータ量対応

IoTやビッグデータなどの大量のデータ保持を効率化

・単一障害点の排除

ELBなどによる高可用設計をしていく

・コスト最適化

リソースのサイズのスケールインスケールアウト

・キャッシュの利用

繰り返し取り出すデータはキャッシュを利用した構成にすること

・セキュリティの確保

全てのレイヤーに対してセキュリティ対応する