可靠性支柱

AWS 良好架构框架

2016 年 11 月

通告

本文档所提供的信息仅供参考，且仅代表截至本文件发布之日时 AWS 的当前产品与实践情况，若有变更恕不另行通知。客户有责任利用自身信息独立评估本文档中的内容以及任何对 AWS 产品或服务的使用方式，任何“原文”内容不作为任何形式的担保、声明、合同承诺、条件或者来自 AWS 及其附属公司或供应商的授权保证。AWS 面向客户所履行之责任或者保障遵循 AWS 协议内容，本文件与此类责任或保障无关，亦不影响 AWS 与客户之间签订的任何协议内容。

目录

摘要

本份白皮书的关注重点在于 AWS 良好架构框架范畴内的可靠性支柱。其中提供的指导将帮助客户在自身 Amazon Web Services（简称 AWS）环境的设计、交付与维护工作当中遵循相关最佳实践。

内容简介

在 Amazon Web Services，我们充分意识到为客户传递架构设计与运营最佳实践的重要意义，特别是在云端可靠性、安全性、效率以及成本效率等层面的积极作用。作为这项工作中的关键性组成部分，我们建立起 AWS 良好架构框架（AWS Well-Architected Framework），其将帮助大家立足 AWS 构建系统的同时，明确把握自身决策中的优势与弊端。我们认为，良好架构系统将能够极大提升成功的机率。

AWS 良好架构框架基于以下五大支柱：

- 安全性
- 可靠性
- 性能效率
- 成本优化
- 卓越操作

本份白皮书着眼于可靠性支柱以及如何将其应用于您的解决方案。在传统内部环境当中，由于存在单点故障、自动化机制缺失以及弹性不足等问题，因此实现可靠性并非易事。通过运用本份白皮书中提及的各项实践，您将能够构建起拥有强大基础、一致性变更管理以及可靠故障恢复流程的架构体系。

本份白皮书适用于各类技术性职能角色，包括首席技术官、架构师、开发人员以及运营团队成员等。在阅读本份白皮书后，您将了解到在设计云架构以实现可靠性方面可资利用的各项 AWS 最佳实践与策略。本份白皮书并不提供具体实施细节或者架构模式信息。不过我们将在文末为您提供与此相关的对应资源。

可靠性

可靠性支柱包括系统从基础设施或者服务中断状况中进行恢复的能力、动态获取计算资源以满足需求的能力以及缓解错误配置或瞬态网络问题等故障的能力。本份白皮书为 AWS 之上的可靠系统架构设计工作提供深入的最佳实践指导意见。

设计原则

在云环境当中，以下几项原则方针能够帮助您切实提升可靠性水平：

- **测试恢复规程**：在本地环境中，测试工作通常用于证明系统在特定情况下的正常运作能力。测试通常不会被用于验证恢复策略。但在云环境下，您可以测试系统在遭遇故障时的具体表现，并借此验证您的恢复规程。您可以利用自动化手段模拟不同故障或重新构建造成早期故障的具体场景。根据由此表现出的问题，您可以在实际故障发生之前作出测试与纠正，并借此降低以往从未测试过的组件可能引发的潜在危害。

- **自动实现故障恢复**：通过监控系统内的各项关键性能指标（简称 KPI），您可以在触及阈值时触发自动响应操作。这种自动化方案中可包含故障通知与追踪机制，以及用于缓解或修复故障问题的恢复流程。凭借着更为复杂的自动化方案，您甚至可以在故障实际发生前作出预测并加以修复。

- **横向扩展以提升整体系统可用性**：利用多种小型资源取代单一大型资源，借此降低单一故障对于整体系统产生的影响。跨越多种小型资源的分布式请求将能够确保其不致共同面对同一故障点。

- **不再依靠猜测确定容量需求**：内部系统当中的一大常见故障根源在于资源饱和，即系统请求的资源超过该系统的资源供给能力（拒绝服务攻击通常正是以此为目标）。在云环境当中，您可以监控系统资源的需求与利用率，并通过自动化方式添加或者移除部分资源以维持其最优水平，从而在无需过度或者过低配置的情况下满足实际需求。

- **自动管理变更**：基础设施内的变更应以自动化方式实现。需要加以管理的一切变更皆应被纳入自动化变更范畴之内。

定义

云环境下的可靠性保护包含以下三项最佳实践：

1. 基础
2. 变更管理
3. 故障管理

为了实现这种可靠性，系统必须拥有经过良好规划的基础且将监控机制部署到位，同时有能力根据需求或者要求处理各项变更。该系统还应在设计当中引入故障检测与自动化自主修复能力。

基础

在构建任何系统之前，您应首先确定可能对其可靠性造成影响的基础性因素。举例来说，您必须拥有充足的网络数据中心传输带宽。此类要求有时会遭到忽略（因为其走出了单一项目的范畴），而这种忽略很可能对系统的可靠性水平造成重大影响。在内部环境当中，由于此类要求可能长期存在且面对极高的依赖度，因此必须在规划初期即被纳入考量。

AWS 设置有多项服务限制（即您的团队所能够请求的每一种资源上限），旨在保护您免受资源过度配置问题的意外影响。您需要部署治理机制与相关流程以监控并变更这些限制，确保其始终与您的业务需求相匹配。在采用云服务的同时，您亦需要思考如何将其同您的现有内部资源加以整合（即混合型方案）。这类混合模式可随时间推移逐步转化为纯云解决方案，因此大家显然有必要在网络拓扑设计当中充分考量 AWS 与内部资源间的交互方式。最后，您还需要确保您的 IT 团队获得必要的培训与更新流程，用以支持您的公有云使用方式，并在必要时引入合作伙伴或技术服务水平协议。

在 AWS 当中，您应在解决基础议题时考量多种不同方案。以下部分阐述了如何实际运用这些方案：

- 限制管理
- 规划网络拓扑结构

限制管理

在进行系统架构设计时，您需要考虑到各项物理限制以及资源限制条件。举例来说，您需要考量光纤连接的数据传输能力或者物理磁盘的具体存储容量。了解物理限制应成为可靠系统设计工作的第一步。另外，在基于服务的架构当中，您需要考虑到用于保障服务水平协议（速率限制）或者设计限制（硬性限制）的具体服务限制因素。限制管理工作中的最后一项要点在于警报与报告。大家必须了解何时可能触及上限或者即将触及上限，并就此作出正确反应。

由 AWS 服务创建的云资源皆存在默认限制，具体条款由各服务进行说明。这些限制可以帐户为单位进行追踪，因此如果您使用多个帐户，则需要了解各个帐户的具体限制水平。其它限制则可能取决于您的实际配置~~方案~~。

以下为资源限制层面的几项常见示例：

- Auto Scaling 组中的 Amazon EC2 实例数量

- 每秒配置输入/输出操作（简称 IOPS）
- Amazon 关系数据库服务（Amazon Relational Database Service ，简称 RDS）存储配额
- Amazon 弹性块存储（Amazon Elastic Block Store ， 简称 EBS）分卷配额
- 网络 IO
- 子网或者虚拟私有云（简称 VPC）中的可用 IP 地址

限制作用于各个 AWS 服务区与各个 AWS 帐户。如果您计划跨服务区部署或者使用多个 AWS 帐户，则应确保增加您所使用之服务区与帐户的限制水平。另外，您还应确保在当前服务区内的可用区中拥有充足的缓冲区容量。这样，一旦该可用区发生问题，您仍可请求额外资源以确保您的缓冲区足以替代一切尚未终止但运行状况不正常的资源。

AWS 通过 AWS Trusted Advisor、AWS 说明文档以及 AWS 管理控制台提供一份服务限制清单与常见问题解答。您可以联系 AWS 支持服务团队以提交当前限制水平。对于用于节流 API 请求的速率限制机制，SDK ~~负责~~提供处理这些节流响应的对应机制（包括重试与指数回退）。您应评估自身用例以选择最适合自身需求的模式。如果节流限制已经对您的应用程序性能造成影响，则应与 AWS 支持服务团队联系以了解是否缓解问题或者增加上限水平。

这一层面中的最佳实践在于自动实现限制追踪。您可以将当前服务限制指标存储在 Amazn DynamoDB 等持久数据存储体系当中。如果您需要将配置管理数据库（CMDB）或者申请系统同 AWS Support API 进行整合，则应确保能够以自动化方式追踪请求增量限制与并发限制。如果您需要整合 CMDB，则可能需要在该系统之内存储具体服务限制信息。

关键性 AWS 服务

判断当前服务限制的关键 AWS 服务为 **AWS Trusted Advisor**，其能够返回一份包含具体限制信息的清单。以下服务亦在其中发挥重要作用：

- **Amazon CloudWatch**：您可以利用 CloudWach 指标设置警报，用以提醒您何时已经接近网络 IO、配置 IOPS、Amazon 弹性块存储（简称 EBS）分卷容量以及临时分卷容量限制。您亦可设定警报以确保在 Auto Scaling 组接近最大容量上限时得到提醒。
- **Amazon CloudWatch 日志**：您可利用指标过滤器对日志事件进行搜索并提取其中的模式。日志条目可被转换为数字化指标，进而配合警报协同运作。

资源

请参阅以下资源以了解限制判断与管理 AWS 最佳实践的相关细节信息。

视频

- 如何管理我的 AWS 服务限制？
- 利用 Amazon SQS 与 Amazon DynamoDB 处理大规模消息（ARC301）| AWS re:Invent 2013

相关工具

- AWS 限制监控器

说明文档

- AWS 服务限制
- 服务限制报告相关博文

规划网络拓扑结构

在利用基于 IP 地址的网络进行系统架构设计时，您需要规划网络拓扑结构并解决未来可能出现的预期增长以及同其它系统与网络进行整合的需求。如果未经过此类前瞻性规划，您可能发现自己的基础设施在扩展及整合方面存在严重局限，或者很难对接非兼容寻址结构。

Amazon 虚拟私有云（简称 VPC）允许您在 AWS 云当中配置一套私有且隔离化分区，从而立足私有网络启动各类 AWS 资源。

当您规划自己的网络拓扑结构时，第一步在于对 IP 地址空间本身作出定义。请遵循 RFC 1918 指南为各 VPC 分配无类别域间路由（简称 CIDR）地址块。另外，在此流程当中亦应考虑以下注意事项：

- 允许 IP 地址空间用于同一服务区内的多套 VPC。
- 考虑跨帐户连接。举例来说，各业务线可能皆拥有彼此独立的帐户与 VPC。这些帐户应能够回接至各共享服务。
- 在 VPC 当中，允许空间支持分布式多个可用区的不同子网。

- 始终在 VPC 之内保留未使用的 CIDR 地址块空间。

网络拓扑结构规划工作的第二步在于确保连接弹性：

- 您如何在拓扑结构当中实现弹性以应对故障状况？
- 如果存在错误配置并移除连接，会引发怎样的影响？
- 您能否处理意外出现的服务流量/使用量增长？
- 您能否抵消拒绝服务（DoS）攻击企图？

AWS 提供的众多功能应在您的实际设计中得到体现。您计划使用多少套 VPC？您是否会在各 VPC 之间使用 Amazon VPC Perring 功能？您是否会将虚拟私有网络（简称 VPC）接入其中任何 VPC？您是否需要使用 AWS Direct Connect 或者互联网？

最佳实践要求您始终根据 RFC 1918 要求为 VPC 无类别域间路由（简称 CIDR）地址块使用私有地址范围。您所选定的范围不应与现有范围或者其它计划通过 VPC Perring 或 VPN 共享的地址空间存在交集。总体而言，您需要确保您所分配的范围中包含远超需部署子网数量、潜在（ELB）负载均衡器大小以及子网内所部署服务器的地址空间。总体来看，您应规划部署大规模 VPC CIDR 地址块。需要 注意的是，VPC CIDR 地址块与子网 CIDR 地址块在部署完成后将无法进行变更。需要注意的是，选择部署最大 VPC 规模可能带来超过 65000 个 IP 地址。基础 10.x.x.x 地址空间意味着您能够实际使用超过 1600 万个 IP 地址。您应确保在决策当中避免发生地址空间过大或者过小的问题。

来自 VPC 的连接由路由表条目负责治理。互联网网关（简称 IGW）、NAT 网关、虚拟私有网关（简称 VGW）或者 VPC Perring 网关皆可作为路由表内的条目供子网使用。当您进行网络结构规划时，请根据需要斟酌使用 VGW 以及 VPC Perring。

在 VPC 之间建立网络的另一种可行方法在于使用 VPN 设备。我们通常建议您选用 AWS Marketplace 当中提供的方案选项。

在选择供应商与实例大小时，您应根据设备运行需求考虑具体弹性与传输带宽要求。举例来说，如果您决定将自己的 VPC 通过 AWS Direct Connect 连接接入自有数据中心，则应利用来自其它供应方的辅助 AWS Direct Connect 连接或者互联网建立冗余连接。如果您采用不具备弹性能力的 VPN 实现方案，则应通过辅助设备建立起冗余连接。对于一切实际场景，您皆需要定义可接受恢复时间（简称 TTR）并通过测试确保其符合您的业务要求。

您应利用现有标准在私有地址空间之内保护各类资源。您应利用单一或者一组子网（每可用 区一套子网）作为互联网与应用程序间的屏障。在内部环境当中，您通常会利用多功能防火墙应对常见网络攻击活动，并利用负载均衡器或者边缘路由器应对 SYN 洪流等拒绝服务（~~简称~~ DoS）攻击。AWS 提供多种服务以实现上

述功能，具体包括 Amazon CloudFront 内的集成 Web 应用防火墙、弹性负载均衡以及 VPC 安全组与网络访问控制列表（ACL）等 AWS 虚拟网络功能。您可以从 AWS 合作伙伴以及 AWS Marketplace 当中挑选符合您实际需求的虚拟设备以实现这些功能。

关键性 AWS 服务

网络规划层面的关键 AWS 服务为 **Amazon 虚拟私有云**（简称 VPC），其允许您分配私有 IP 地址范围以实现无互联网资源访问或者数据中心延伸。以下服务与功能亦在其中发挥重要作用：

- **AWS Direct Connect**: AWS Direct Connect 可用于提供面向 AWS 的私有专用连接，借以实现更低延迟与更具一致性的 AWS 往来性能。
- **Amazon 弹性计算云（ Elastic Compute Cloud ，简称 EC2）**：如果您选择在各网络之间采用 VPN，则可立足 Amazon EC2 运行 VPN 方案。
- **Amazon Route 53**: Amazon Route 53 属于一项域名系统（简称 DNS）服务，其可直接与 ELB 相集成并在发生拒绝服务（简称 DoS）攻击时提供防御层。
- **弹性负载均衡（简称 ELB）**：ELB 可跨越多个可用区提供负载均衡能力，并可配合 Auto Scaling 实现基础设施的自我修复能力。

资源

请参阅以下资源以了解更多与网络规划 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2015 | (ARC403) 由一到多：VPC 设计演进
- AWS re:Invent 2015 | (SEC306) 抵御 DDoS 攻击

说明文档

- Amazon 虚拟私有云产品页面
- Amazon 虚拟私有云说明文档
- AWS 问题解答中的 Transit VPC 文章
- Amazon EC2 实例类型产品页面

- Amazon EC2 实例类型说明文档
- AWS Marketplace 网络基础设施
- DDoS 弹性 AWS 最佳实践白皮书
- Amazon VPC 连接选项白皮书

变更管理

了解变更对于系统的实际影响能够确保您主动作出规划，而监控能力则可帮助您快速发现一切可能导致容量问题或者 SLA 违规的负面趋势。在传统环境当中，变更控制流程通常以手动方式进行，且必须配合认真协调与审计方可有效控制执行操作的具体个人及活动时间。

利用 AWS，您将能够监控系统活动并自动根据 KPI 作出响应——例如在系统迎来更多用户时自动添加服务器数量。您可以控制哪些用户拥有权限以作出系统变更，或者对系统变更记录进行审计。

在 AWS 当中，您可利用多种不同方案实现变更管理：

- 按需变更
- 监控
- 变更执行

按需变更

产品人气最高之时，往往也正是可靠性层面暴露具体问题的高危阶段。通过确保拥有充分余量满足需求，您将能够有效应对各类组件故障。高需求测试机制对于成功实现可靠性至关重要。

所谓**弹性**，是指云环境能够提供几乎无限资源容量，并由云服务供应商负责资源容量的管理与配置工作。通过使用 API，您可以通过编程方式动态调整架构当中的云资源规模。如此一来，您即可横向扩展架构内的各组件规模，同时在发生需求峰值时自动增加资源数量以始终维持高可用性；并在需求回落时清退资源以避免成本浪费。

在 AWS 当中，上述目标主要通过 Auto Scaling 实现，其可帮助您根据预定义条件自动进行 Amazon EC2 容量的规模伸缩。Auto Scaling 通常配合弹性负载均衡（简称 ELB）共同使用，从而立足单一 Auto Scaling 组跨越多个 Amazon EC2 实例进行输入应用流量分发。Auto Scaling 利用规模伸缩计划进行触发，此计划中包含规

模伸缩策略定义（包括手动、计划内或者按需）以及可在 Amazon CLoudWatch 内监控的各项指标与警报。CloudWatch 指标可用于触发规模伸缩事件。您可利用 CPU 利用率、负载均衡器观察到的请求/响应延迟等标准化指标，亦可使用由 EC2 实例内应用代码建立的定制化指标。

Auto Scaling 亦可在无需负载均衡器或 CloudWatch 指标的情况下独立使用。举例来说，您可以设置一组工作程序以处理队列，或利用定制化代码触发 Auto Scaling 对特定业务活动或者竞赛日活动的响应。

在您进行架构设计时，请务必关注以下两项与弹性相关的注意事项：首先，您能够以怎样的速度实现新资源配置？其二，您需要在供需之间保留怎样的余量，从而确保有能力应对需求量变化或者资源故障？

您可以通过减少实例引导时的启动项与配置任务数量优化配置速度。这项工作通常可通过预配置 Amazon Machine Image（简称 AMI）以“预置模式”实现。需要注意的是，这类优化方式可能牺牲实例的可配置性，因此您需要根据对速度以及配置灵活性的实际需求作出权衡。供需之间的余量在初始开发时往往相当可观，但随着您测试与生产工作的不断推进，您可通过实验逐步降低余量水平并建立信心。

在 AWS 当中，您可利用预热机制以处理已知条件下的快速容量变化。应用程序功能之外的预填充缓存或者集群及/或实例组向外扩展等皆可进行预热或者以编程方式进行初始化：

- **Amazon 弹性块存储（ Amazon Elastic Block Store，简称 EBS）**： 可利用快照实现。您能够在接收负载之前读取全部块内容以降低首次访问存储块时的 IO 操作延迟。
- **Amazon DynamoDB**: IO 可通过配置表以编程方式进行变更，具体包括设置以及读取/写入容量单元。
- **Amazon 关系数据库（Amazon Relational Database，简称 RDS）**：面向高强度读取数据库工作负载的向外扩展读取副本。
- **Amazon API Gateway**: 尽管仍受到帐户限制影响，您可利用 REST API 对阶段级或方法级节流限制作出设置。
- **AWS Lambda**: 根据您提使用的具体语言，AWS Lambda 可能需要配合特定途径以在容量需求迅速增长的情况下预先“启动”对应容量。

您应在基础设施即代码体系之内遵循各项软件开发最佳实践。一切变更皆应立足多个层级接受测试。您应始终以尽可能接近实际条件的情况下对实现方案进行测试，而后方可将其自动部署至生产环境当中。欲了解更多相关信息与最佳实践，

请参阅《AWS 良好架构框架性能效率支柱》白皮书当中的“负载测试与基准测试”章节。

关键性 AWS 服务

自动化按需管理层面的关键 AWS 服务为 **Auto Scaling**，其允许大家定义多个 Auto Scaling 组并与 CloudWatch 警报机制相集成以触发规模伸缩操作。以下服务与功能亦在其中发挥重要作用：

- **Amazon CloudWatch 事件**：此类事件可根据特定控制方案执行复杂的规模伸缩操作。
- **AWS Elastic Beanstalk**: AWS Elastic Beanstalk 可为您配置 Auto Scaling 与弹性负载均衡服务。其亦可管理操作系统以及各应用程序容器。
- **AWS OpsWorks**: AWS OpsWorks 负责提供自动化代码，用以帮助您根据时间或者负载进行应用程序规模伸缩。

资源

请参阅以下资源以了解更多与自动按需管理 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2015 | (DVO304) AWS CloudFormation 最佳实践
- AWS re:Invent 2014 | (ARC317) 立足超大规模（Scryer）维护弹性门户

说明文档

- Auto Scaling 说明文档
- Amazon CloudWatch 事件说明文档
- AWS Elastic Beanstalk 说明文档
- AWS OpsWorks 说明文档
- EBS 分卷初始化说明文档
- 以规模化方式管理您的 AWS 基础设施白皮书

监控

缺少细化的监控与警报机制作为支持，变更管理工作将无法实现。尽管大家已经相当熟悉操作系统之内的监控工作，但云环境中的监控机制则带来一系列新的机遇。相较于利用 SNMP 等传统标准化方案，云服务供应商开发出更多定制化方案，足以钩接从实例性能到网络层再到请求 API 本身的一切对象并借此提取观察结论。

AWS 环境下的监控工作主要分为以下五个阶段：

1. 分代
2. 聚合
3. 实时处理与警报
4. 存储
5. 分析

分代

首先确定哪些服务与/或应用程序需要进行监控，定义相关重要指标，同时考量如何在必要时从日志条目中提取这些指标以建立阈值及警报事件。AWS 提供丰富的监控与日志信息供您使用，您亦可利用其定义按需变更流程。以下为能够生成日志及指标数据的部分服务与功能列表 ：

- Amazon EC2 容器服务（Amazon EC2 Container Service，简称 ECS）、Amazon EC2、经典负载均衡器、应用程序负载均衡器、Auto Scaling 以及 Amazon EMR，可发布面向 CPU、网络 IO 以及磁盘 IO 均值的指标。
- Amazon CloudWatch 日志适用于 Amazon 简单存储服务（Amazon Simple Storage Service，简称 S3）、经典负载均衡器以及应用程序负载均衡器。
- Amazon VPC 工作流日志适用于 VPC 环境下的任意或一切弹性网络接口（简称 ENI）。
- AWS CloudTrail 以单一帐户为基础记录一切 API 事件。
- Amazon CloudWatch 事件可交付一套实时系统事件流，用以描述 AWS 服务中的各项变更。
- AWS 提供相关工具以收集操作系统级日志，并以数据流形式将其导入 Amazon CloudWatch 日志。
- 定制化 Amazon CLoudWatch 指标可用于建立任意维度下的对应指标。

- Amazon ECS 与 AWS Lambda 可将日志以数据流形式导入 CloudWatch 日志。

聚合

Amazon CloudWatch 与 Amazon 简单存储服务（简称 S3）可作为一级聚合与存储层。对于 Auto Scaling 及 ELB 等部分服务，AWS 提供 CPU 负载以及跨集群或实例平均请求延迟作为“开箱即用”的默认指标。而对于 Amazon VPC 工作流日志或者 AWS CLoudTrail 等流式服务，事件数据则将被转发至 Amazon CloudWatch 日志当中，而您则需要定义并应用过滤机制以从事件数据内提取指标。您将能够借此获取时间序列数据并定义 CloudWatch 警报阵列以触发对应警报。

实时处理与警报

警报可触发 Auto Scaling 事件，以便集群根据需求作出对应变更。警报亦可发送至各 Amazon 简单通知服务（Amazon Simple Notification Service，简称 SNS）主题，并随后被推送至任意数量订阅方处。举例来说，Amazon SNS 能够将警报转发至邮件昵称处，确保技术支持人员能够及时作出响应。另外，警报亦可被发送至 Amazon 简单队列服务（简称 SQS）处，此服务可作为面向第三方申请系统的集成点。最后，AWS Lambda 也能够进行警报订阅，从而为用户提供一套能够动态反应各类变化的异步式无服务器模式。

存储与分析

Amazon CloudWatch 日志还能够支持订阅机制，意味着其允许数据以无缝化方式被导流至 Amazon S3。当 CloudWatch 日志及其它访问日志抵达 Amazon S3 时，您应考虑利用 Amazon EMR 从数据当中进一步提取见解与价值。

各合作伙伴及第三方厂商同样提供大量工具选项，旨在帮助您完成聚合、处理、存储以及分析等任务。其中部分卓越工作代表包括 New Relic、Splunk、Loggly、Logstash、CloudHeatlh 以及 Nagios 等等。不过，系统与应用程序之外的日志往往因云服务供应商而异，且各项服务通常拥有属于自己的日志形式。

监控流程当中常常受到忽视的一大重要因素为数据管理。您需要确定监控数据的保留需求，而后据此引入生命周期政策。Amazon S3 在 S3 存储桶层级支持生命周期管理机制。这一生命周期管理方案可通过不同路径引入存储桶当中。在生命周期结束之后，您可以将数据移到地 AWS Glacier 进行长期存储，直到数据内容彻底过期。

关键性 AWS 服务

监控层面的关键 AWS 服务为 **Amazon CLoudWatch**，其允许您轻松创建警报并借此触发各类规模伸缩操作。以下服务与功能亦在其中发挥着重要作用：

- **Amazon S3**：作为存储层存在，且允许配合生命周期政策及数据管理方案共同使用。
- **Amazon EMR**：您可利用此服务从日志及指标数据中获取更深层次的见解。

资源

请参阅以下资源以了解更多与监控 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2015 | (DVO315) 利用 Amazon CloudWatch 对您的 IT 体系进行记录、监控与分析
- AWS re:Invent 2015 | (BDT312) 应用程序监控：数据情景为何如此重要

说明文档

- Amazon CloudWatch 说明文档
- Amazon CloudWatch 日志说明文档
- View Amazon EMR 日志文件说明文档

变更执行

凭借基础设施即代码机制，变更管理能够以软件开发的形式实现。基础设施当中的变更可描述为运行环境与源控制内现有对象之间的“**差异**”。您可以设置开发、测试及生产环境，确保能够以高效方式在实际部署之前对变更进行测试。您亦可立足这些环境测试任何新型应用程序或者软件的整体部署流程。在非云环境下，大家几乎不可能建立起等同于生产环境的开发与测试环境。但利用云资源，您将能够确切部署与内部生产设施相同的网络、防火墙以及入站与出站路径。二者之间的差异在于执行部署所需要的具体权限。您可以拥有相同的日志聚合服务及监控等因素。您的全部自动化机制皆应适用于一切环境。如此一来，由于您已经成功完成了开发及测试环境的部署，因此能够显著降低生产环境部署所面临的潜在风险。

部署的实现方式多种多样，具体包括直接部署、蓝/绿（或者红/黑）部署以及金丝雀部署等。利用自动化方式实现的应用程序新版本部署将成为一种常态运营任务，而非变更任务。而您所需要管理的变更现在以自动化方式完成，且具体执行方式遵循您的现有变更管理流程。

在 AWS 当中，实现基础设施即服务的关键性服务为 AWS CloudFormation。您可以将各类基础设施与应用程序组成部分作为独立 AWS CloudFormation 堆栈进行部署。您的堆栈应采取松散耦合状态且尽可能保持简单。您可以利用 AWS 软件开发套件（简称 SDK）配合您所熟悉的语言自动完成部署工作。

您可以利用其它 AWS 服务辅助整个部署流程。AWS CodeDeploy 允许您自动执行系统部署流程中的各项步骤。AWS CodePipeline 可用于实现系统的持续交付。而 AWS OpsWorks 则采用 Chef，意味着频繁使用 Chef 的客户能够直接利用其现有 Chef“食谱”。AWS Elastic Benstalk 则允许您立足一套负载均衡且自动规模伸缩环境轻松完成部署，您无需在其中维护任何操作系统或者语言运行时容器。AWS Service Catalog 允许您创建标准化 AWS CloudFormation 模板，而后面向内部用户进行分发。AWS Config 规则则可识别各项变更是否遵循合规性要求，并据此加以执行。

您可以利用 AWS SDK 实现蓝/绿或者金丝雀部署。AWS 还发布了一份专项白皮书，旨在探讨蓝/绿部署方案中的各类主流方法。此份白皮书在本文文末的参考部分内给出链接。一般来说，您可能需要采用多种部署方法，具体取决于您所使用的技术与已经部署的系统。您还可以咨询 AWS 交谊架构师以了解如何保证这些技术方案切实契合您的现有流程，或者如何定义新型流程。

关键性 AWS 服务

支持基础设施即代码方法的关键 AWS 服务为 **Amazon CloudFormation**，其允许您利用等同于管理部署代码的方式管理自己的基础设施与系统。以下服务与功能亦在其中发挥重要作用：

- **AWS CodePipeline**：允许您实现系统持续交付。
- **AWS CodeDeploy**：允许您立足现有系统进行代码部署。
- **AWS Elastic Beanstalk**：允许您轻松部署由多种语言编写而成的代码，且无需管理操作系统或者语言容器。
- **AWS OpsWorks**：允许您利用现有 Chef“食谱”在 AWS 上部署自己的应用程序。
- **AWS 身份与访问管理（简称 IAM）**：允许您管理各类系统中使用的面向 AWS API 的访问与权限。

- **AWS Service Catalog**：立足于 AWS CloudFormation 构建而成，旨在帮助您面向内部客户发布模板。
- **AWS Config**：允许您跨越多种变更与资源集执行合规性要求。

资源

请参阅以下资源以了解更多与基础设施即代码 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2015 | (DVO304) AWS CloudFormation 最佳实践
- 利用 AWS 加速 DevOps 管道

说明文档

- AWS 上的蓝/绿部署白皮书
- AWS CloudFormation 说明文档
- AWS 工具与软件开发套件产品页面
- AWS CodePipeline 说明文档
- AWS CodeDeploy 说明文档
- AWS 身份与访问管理说明文档
- AWS OpsWorks 说明文档
- AWS Elastic Beanstalk 说明文档
- AWS Service Catalog 说明文档

故障管理

在任何复杂系统当中，我们都有必要作好发生潜在故障的预期，更重要的是应了解如何发现此类故障、作出响应并防止其再次出现。

在 AWS 当中，您可以利用自动化优势响应监控数据。举例来说，当某一特定指标触及阈值时，您可以触发自动操作以解决该问题。另外，相较于尝试对生产环境中的部分故障进行诊断与修复，大家可以直接利用新资源加以取代并对故障资

源进行分析。由于云环境能够以低成本方式为您提供完整的临时性系统版本，因此您完全能够采取自动化测试机制验证整个恢复流程。

在 AWS 当中，大家可以利用多种不同方式考量故障管理问题：

- 数据持久性
- 承受组件故障
- 恢复规划

数据持久性

数据丢失可以算是任何系统当中最为可怕的故障模式。对于业务连续性规划而言，企业需要定义一项可接受的恢复点目标（简称 RPO），其负责定义发生故障时数据丢失的持续时长。另外，大家还需要定义恢复时间目标（简称 RTO）以设定实现功能恢复所需要的时长。持久性保障工作还涵盖您的备份数据持久性。在实际定义当中，具体时长需要符合可行性与成本效益，且取决于不同系统与数据的特定情况。

最佳实践要求定期通过数据的恢复与验证对您的备份规程进行测试。您应证明在数据恢复过程中维持自身正常运营的能力，且不依赖于任何受影响情景。这一层面中的常见问题包括缺乏访问文档、软件、磁带设备以及加密密钥的能力。指向备份数据的访问权限应以等同于数据安全访问的水平加以治理。

Amazon S3 原生提供 99.999999999%（十一个九）持久性。AWS 服务快照存储于 Amazon S3 当中。当您使用 Amazon EBS 时，我们建议您频繁保存快照以提升持久性水平。如此一来，您将能够在故障发生时尽可能以更新快照执行恢复。初始 Amazon EBS 快照为完整镜像，而此后的各快照则属于增量式快照，旨在降低实现成本。Amazon RDS 需要进行定期快照保存，而您则可根据需要选用各时间点快照。初始 Amazon RDS 快照作为基准，此后的增量式快照以其为基础，借此尽可能降低存储空间占用量。

其它拥有持久存储机制的 AWS 服务还包括 Amazon 弹性文件系统（简称 EFS）、Amazon SQS、Amazon Kinesis 以及 Amazon DynamoDB。这些服务皆以自身组成部分的方式提供数据持久性保障。AWS 密钥管理服务（AWS Key Management Service ，简称 KMS）可帮助您管理指向加密密钥的访问，同时管理这些密钥的具体生命周期。您可以弃用数据加密用密钥，但继续利用这些密钥进行数据解密，直到数据因生命周期结束而被删除，此后相关密钥亦将被清空。

您可在 Amazon EBS 分卷之上采用软件 RAID，从而在无需定期保存快照的前提下实现更高持久性。然而，这种作法会带来更高支出，因为您需要使用更多 Amazon EBS 分卷。另外，这种方案亦可能导致性能退休，因为每项事务都会引发面向多块磁盘的写入操作。配置 IOPS 能够帮助您解决此类问题（但同样会产生额外费

用），但主要适用于对于持久性要求最高的高性能系统。如果您在实例之上使用临时性存储机制，则需要利用备份软件为应用程序保存快照。

着眼于持久性保障，您在选择数据存储方案时的主要考量在于：持久性要求、一致性快照获取难度、一致性快照获取速度以及快照恢复速度。

关键性 AWS 服务

持久性备份策略层面的关键 AWS 服务为 **Amazon S3**，其可提供极高水平的数据持久性。以下服务与功能亦在这一层面发挥重要作用：

- **AWS KMS**：允许您轻松配合其它 AWS 服务管理加密密钥。
- **Amazon EBS**：提供一项面向块存储的托管服务，且可用于各 EC2 实例。

资源

参考以下资源以了解符合 AWS 最佳实践的存储方案选择以及如何利用服务内存储资源的相关细节信息。

视频

- AWS re:Invent 2014 | (SOV203) 了解 AWS 存储选项
- AWS re:Invent 2015 | (SEC301) 利用 AWS 内加密机制保护数据的相关策略

说明文档

- Amazon S3 说明文档
- AWS KMS 说明文档
- Amazon EBS 说明文档
- Amazon RDS 说明文档
- Amazon DynamoDB 说明文档
- Amazon SQS 说明文档
- AWS Kinesis 说明文档

承受组件故障

当考量架构体系的可靠性时，理解并消除其中的单点故障因素是实现目标的关键所在。我们通常需要采取**负载分担方式**，其中将利用额外资源分担负载以缓解单点故障问题。这种负载分担机制适用于全部层级——从服务器到数据中心皆涵盖于其中。

在云环境当中，最佳实践要求定义一项恢复平均时间（简称 MTTR），而非单纯为各组件定义平均故障间隔时间（简称 MTBF）。云环境能够利用自动化机制交付执行所需要的容量，且在修复过程中向相关人员发布通知。

无状态系统可通过增加额外的同等资源（即**横向规模伸缩**）实现规模伸缩。通过将横向规模伸缩与负载均衡机制相结合，您能够确保同类负载被分发至各套横向实现方案之内。当您希望构建起有能力应对服务故障的系统及软件时，实际上相当于提升系统在恢复过程中承受故障的能力。您应对系统进行解耦，以避免产生连锁性影响。经过解耦的系统将与关联系统故障彼此隔离开来，从而在恢复执行过程中继续保持正常运行。

云环境能够轻松实现横向规模化系统的部署与自动化能力。AWS 提供多项服务，用以实现负载均衡、队列、通知发送、自动化恢复以及多位置使用等能力。

AWS 提供众多功能与服务，具体包括多服务区内的多可用区、经典弹性负载均衡器、应用程序负载均衡器、Amazon SQS、Amazon SNS 以及 Auto Scaling 组等等。一部分 AWS 服务还天然拥有极高容错性，包括 Amazon S3、Amazon DynamoDB、AWS Lambda 以及 ELB 等等。此外，AWS 服务还可通过应用程序编程接口（简称 API）实现可用性，允许您以自动化方式实现更为全面的容错能力。

ELB 提供的运行状态检查使您能够报告资源状态，确保服务本身能够随时替换运行状况异常的资源。这些运行状态检查应采取等同于您正常运营方式的机制，例如指向 Amazon RDS（数据层）——而非 Amazon RDS API 调用（控制层）——的数据库队列。运行状态检查应能够报告资源当中的问题位置，并通过资源替换修复该问题。如果问题由下游依赖关系所造成，则您的运行状态检查则不应报告问题——因为这有可能导致资源发生不必要的中断，而非修复依赖关系这一真正根源。

关键性 AWS 服务

- 承受组件故障相关测试层面内的关键 AWS 工具为 **AWS SDK**，其允许大家以自动化方式实现故障恢复并模拟故障场景。以下服务与功能在这一层面中同样发挥重要作用：

- **ELB**：跨越多个实例的负载均衡机制允许您在发生实例故障时继续正常交付服务，监控实例运行状态并自动替换发生故障的实例。
- **Amazon SQS**：允许您将操作从请求中解耦出来，同时对由请求生成的流程依赖关系进行规模伸缩。
- **Amazon SNS**：允许您利用事件驱动型通知模式执行各项操作。
- **Auto Scaling**：允许您根据需要添加及移除资源。

资源

请参阅以下资源以了解更多与容错性系统构建 AWS 最佳实践相关的细节信息。

视频

- 故障中的弹性机制——Netflix 的终极云端可用性保障方案
- AWS re:Invent 2014 | (ARC309) 构建并扩展面向数百万用户的云驱动体系
- AWS re:Invent 2014 (PFC305) 拥抱故障：故障注入与服务弹性
- AWS re:Invent 2015 (ARC401) 云至上：新型基础设施中的新型架构

说明文档

- AWS 开发者工具与 SDK
- AWS 全球基础设施
- Amazon SQS 说明文档
- Auto Scaling 说明文档
- 弹性负载均衡说明文档
- Amazon SNS 说明文档
- 云环境下的基准可用性与可靠性

恢复规划

当组件出现故障或者大规模中断状况影响到您对于系统的操作能力时，最重要的无疑是了解接下来可能发生的各类问题。您需要制定灾难恢复计划，并确保其得到充分理解、证明与贯彻。另外，您架构体系的弹性水平也必须得到充分测试。

弹性测试：您的常规开发流程应涉及组件测试以及故障恢复相关规程。在生产环境当中，故障状况通常由多个组件间的交互所共同引发。因此，您应当以生产规模对架构进行全面测试。您应在测试流程当中检查架构的具体故障原因，并据此找到可先找恢复手段。从历史角度来看，恢复流程往往是系统测试当中最不受重视的部分。然而，您绝对有必要建立起一套负责定义故障后处理方式的响应剧本。另外，利用故障注入手段进行异常场景模拟。当生产环境中发生故障时，您可采用根本原因分析（简称 RCA）流程以从失败中总结经验，并减轻未来可能出现的类似状况。最后，您应规划**竞赛日活动**，在企业之内执行大规模测试——这不仅是为了验证您的架构体系，同时也是为了确保您的运营手册及 RCA 流程能够在非常时期真正解决问题。

灾难恢复

您需要测试自己的灾难恢复流程与规程，借此确保其正常运作能力以及各团队知晓如何加以使用。定期演练以确保在事故实际发生时，各流程与规程皆可得到正确理解及执行。您发现的任何问题都应被明确记录在文档当中，同时说明其根本原因并保证采取相关步骤加以纠正。如果可能，这部分信息还应被纳入广泛的共享知识库当中，从而防止类似的错误在其它系统当中再次发生。

更重要的是，您灾难恢复站点中使用的各版本应与主站版本完全一致。如果两套站点之间未能确切同步，则会造成所谓“**漂移**”问题。避免漂移的理想方式在于确保您能够利用自动化机制将各套方案的最新版本发布至所有位置处。您的灾难恢复流程应包含业务保护规划，包括其在恢复模式下的运行周期。这套规划应列举具体步骤，用以确保您的主站点在恢复正常之后，能够切实同步此阶段内恢复站点接收到的更新数据。

AWS 云立足于多个服务区与可用区（简称 AZ）构建而成。一个服务区代表世界范围内的某一物理位置，我们在其中建设有多个可用区。可用区由一座或者多座离散数据中心组成，其中各数据中心皆拥有独立的冗余电源、网络与连接，且全部安装在单独设施之内。我们的可用区能够为您提供远超单一数据中心的可用性、容错性以及可扩展性水平。文末的资源章节内列出了 AWS 灾难恢复白皮书，其中总结了面向内部系统乃至 AWS 云环境的多种灾难恢复实现方案。

在 Amzon S3 之内存储关键性文件时，您应启用版本控制机制，同时在 IAM 实体之内禁用版本删除权限，并利用对象生命周期管理政策以治理数据删除操作。这将确保您不致遭遇数据被意外或者恶意删除的状况。您应利用另一个 AWS 帐户作为灾难恢复机制，同时建立跨帐户权限以确保主 AWS 帐户只可用于添加内容，而不可删除内容。对于第二帐户，您需要确保作用于主帐户的一切限制同样被纳入此灾难恢复帐户。AWS 提供 API 以实现 Amazon Machine Images（简称 AMI）、

Amazon EBS 快照、Amazon RDS 快照以及 Amazon DynamoDB 流等多种对象的资源共享。

利用 AWS 服务，您可以自动将全部系统交付至另一 AWS 服务区或者帐户当中。您随后即可使用这些新版本。自动测试机制允许您组织竞赛日活动，用以验证您的灾难恢复规程并对灾难恢复方案进行负载测试。如果您选择采用主动-被动故障转移配置方案，则可利用 Amazon Route 53 运行状态检查机制将流量重新定向至灾难恢复服务区或者帐户，并在主服务区或帐户恢复正常后重新恢复流量路由目的地。

您应确保您的自动化体系内包含充足的抽象因素，从而允许其运行在各 AWS 服务区与帐户当中。您还需要确保您的数据访问与保留策略切实有效。

在选择灾难恢复模式时，请考虑以下注意事项：

- 常用功能（例如部署、调查及数据上传等）跨帐户访问管理所带来的复杂性水平。
- AWS 密钥管理服务内基于服务区的密钥所产生的影响。快照本身并不允许您利用来自其它服务区的密钥恢复加密数据快照。
- 运行状态检查的有效性。您需要确保不致因单一子系统内发生的瞬态错误而错误触发灾难恢复流程。
- 如果使用第二服务区，则应考虑到数据复制延迟。数据复制至另一服务区内的具体速度将对您的 RPO 造成直接影响。

关键性 AWS 服务

灾难恢复方案支持层面的关键 AWS 服务为 **AWS 身份与访问管理**（简称 IAM），其允许您治理灾难恢复流程当中的各类必要数据访问操作。以下服务与功能亦在此层面中发挥重要作用：

- **Amazon S3**:此项服务的版本控制、生命周期管理、跨服务区复制以及高持久性存储能力使其成为数据存储中的关键性服务选项。
- **Amazon Glacier**: Glacier 是一项与 Amazon S3 密切相关的归档数据存储服务。
- **AWS 密钥管理服务**: 了解 AWS KMS 密钥的区域特性对您利用 AWS KMS 进行快照加密与解密的能力至关重要。
- **Amazon Route 53**: 提供运行状态检查功能，您可借此发现故障并协助实现灾难恢复规程。

资源

请参阅以下资源以了解更多与灾难恢复 AWS 最佳实践相关的细节信息。

视频

- AWS re:Invent 2014 | (BAC404) 利用 AWS 部署高可用性与灾难恢复架构
- AWS re:Invent 2014 (SDD424) 利用 DynamoDB 流简化可扩展分布式应用程序

说明文档

- Amazon S3 对象版本控制说明文档
- Amazon S3 对象生命周期管理说明文档
- Amazon S3 跨服务区复制说明文档
- AWS KMS 说明文档
- Amazon Route 53 运行状态检查说明文档
- 利用 AWS 实现备份、归档与恢复方案白皮书
- AWS 灾难恢复白皮书

结论

可靠性是一项持续性工作。当故障发生时，我们应将其视为提升架构可靠性水平的重要机遇。定期测试故障发生原因，您将能够更为主动地在其实际出现之前加以纠正。如果生产环境中发生故障，经过良好测试的架构体系则能够有效缓解相关影响。凭借着各项 AWS 服务及其强大的编程功能，您将能够更轻松地建立起灾难恢复解决方案。

AWS 致力于通过提供具备理想弹性水平、响应能力以及适应性的系统帮助您实现价值。为了确保架构真正具备弹性，您应充分遵循本份白皮书中提及的各项最佳实践。

贡献者

以下个人及组织为本文的编撰工作作出贡献：

- Philip Fitzsimons，Amazon Web Services 高级良好架构经理
- Rodney Lester， Amazon Web Services 首席专业服务顾问
- Michael Wallman，Amazon Web Services 高级专业服务顾问

扩展阅读

欲获得更多帮助，请参阅以下资料：

- AWS 良好架构框架