AWS 云采用框架

流程论点

2016 年 2 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

Amazon Web Services（简称 AWS）云采用框架（简称 CAF）[1]提供各项最佳实践与规范性指导，帮助企业客户加快迈向云计算的转型步伐。本份 CAF 指南分为多个着眼领域，且全部与基于云的 IT 系统实现工作密切相关。这些重点领域被称为“论点”。每项论点由一份单独的白皮书负责论述。本份白皮书论述的“观点”方向为成熟度，其核心内容在于评估企业客户的现状、确定未来状态，同时创建路线图以勾勒采用云 IT 功能后所带来的效果。

简介

流程论点涵盖云采用当中 IT 生命周期内的各相关举措。其着眼于在于将 IT 项目作为组合加以管理，从而实现投资优化、确保所交付服务符合质量目标以及通过经过良好定义的计划与项目承载工作内容等预期。在云软件开发领域，您可以使用敏捷性方法与迭代化生命周期，旨在提供增量式功能升级并率先发现并修复各类缺陷。您亦可以使用持续集成/持续交付（简称 CI/CD）实践，从而通过自动化方式完成软件的构建、测试与部署。另外，您可以使用 CI/CD 方法自动完成运维流程，并借此显著提升解决方案弹性并降低手动操作工作量。图二所示为宏观层面的流程论点组成部分、举措以及实施手段等重要内容。

图一：流程论点中的各组成部分

AWS CAF 流程组成部分	举措	实施手段
投资组合管理	▪ 为决策制定定义经济框架 ▪ 缩短规划周期 ▪ 利用面向目标的工程技术要求实现目标状态	▪ 企业 IT 领域示意图 ▪ 企业 IT 功能到企业领域矩阵 ▪ 包含延期成本的经济框架 ▪ 影响图
服务交付管理	▪ 调整现有服务交付流程 ▪ 发起精益-敏捷工程实践 ▪ 提出 DevOps 文化发展目标	▪ 精益-敏捷工程技术 ▪ DevOps 实践
计划与项目管理	▪ 建立一套组合，用于管理全部业务与 IT 功能 ▪ 将基于云的 IT 服务整合至现有采购模式当中	▪ 企业组合整理表 ▪ 采购模式 ▪ AWS 应用迁移方法 ▪ AWS 迁移工厂（AWS Migration Factory）
持续集成与持续交付	▪ 利用文档记录当前 SDLC 实践 ▪ 为团队开发方案以通过持续改进实践采用 CI/CD 实践 ▪ 实现交付通道	▪ CI/CD 工具矩阵
流程自动化	▪ 通过文档利用价值流程图或者其它技术记录当前流程 ▪ 整理当前流程并进行优先级排序，从而实现优化 ▪ 整理当前流程并进行优先级排序，从而实现自动化	▪ 价值流程图 ▪ 基于举措的会计核算
质量管理	▪ 发起持续改进实践 ▪ 将持续改进机制整合至全部 IT 实践当中	▪ 价值流程图 ▪ 约束理论 ▪ 根本原因分析 ▪ 改进套路（Improvement Kata） ▪ 指导套路（Coaching Kata） ▪ 规划、实施、检查、执行（简称 PDCA）

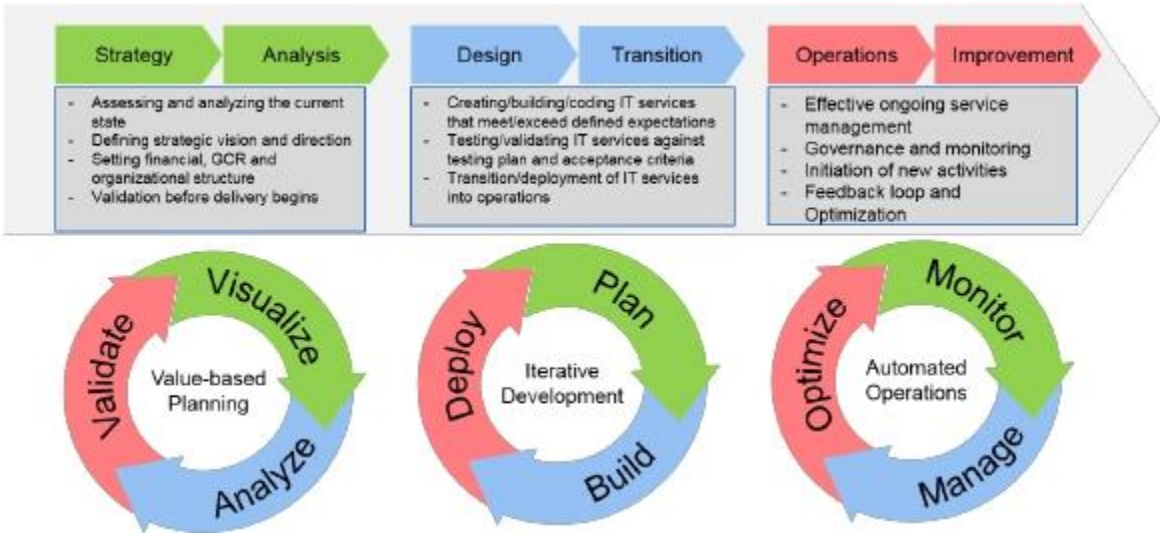
图二：流程论点中的举措与实施手段

服务交付管理

作为 AWS CAF 流程观点中的组成部分，服务交付管理负责推广人员、流程以及技术元素的使用方式，从而帮助大家优化业务成果交付效果。您应当考虑利用以下方案充分发挥云 IT 服务中的既有优势，从而优化自身服务交付流程：

1. 利用文档记录您的当前服务交付流程。
2. 在云战略开发过程当中，对已经定义的目标成果进行审查。
3. 确定支持云战略内所定义成果的最低流程修改要求。
4. 在掌握了所需目标状态的实际情况后，开始通过内部变更管理流程实际着手进行变更。

图三所示为 IT 生命周期以及审查目标状态时各关键性领域的相关示例。



图三：IT 生命周期

此套方案能够确保大家以最小投入成功将云 IT 功能纳入现有服务交付流程。

利用 DevOps 对业务进行改造

要通过云 IT 服务采用举措实现最佳价值，大家应当考虑同时采用精益敏捷工程实践与 DevOps 文化理论。正如 Damon Edwards 所言，“DevOps 的全部诉求在于帮助企业尽快、尽可能高效且尽可能可靠地对各类市场因素做出反应。”[2] 之所以能够带来这样的效果，是因为 DevOps 专注于优化服务价值流程，包括从构思到客户消费的每一个环节。正是这一改造类型使得 Amazon 及众多其它企业得以立足于不断增长的业务规模始终保持创新能力。

谁构建，谁运行

在云 IT 交付模式中使用 DevOps 理论能够带来诸多助益。AWS CTO Werner Vogels 曾将其归纳为“谁构建，谁运行”原则。在 2006 年的采访当中，Vogels 描述了由各团队独立开发并运营服务的原有模式转型至单一团队既负责开发、又负责运维的新型模式所能带来的实际价值。“小型团队概念意味着大家在执行过程 中，能够凭借持续反馈回路确实了解现有方案给客户带来的影响。”[3]

考虑利用以下改进举措在现有 IT 运营模式下实现 DevOps 价值：[4]

- **面向生产的设计**—“谁构建，谁运行”原则使得开发团队必须认真考虑其软件成果如何在生产环境下运行，且运行效果是否与设计目标相符。这能帮助您的团队避免在开发末期才意识到生产环境中存在会导致现有方案无法顺利起效的意外因素。事实上，这种状况相当常见且会给软件质量造成极大危害。一旦出现此类问题，我们必须在部署周期之内做出必要调整以弥合生产与开发之间的条件差异。然而在后续测试当中，大家很可能发现这些调整会在系统的其它位置造成意料外的错误。

- **鼓励提升员工自主权**—“谁构建，谁运行”心态会鼓励软件成果的归属关系与连带责任，从而培养出更多更加独立、有责任心且具备职业成长潜力的企业员工。

- **进一步提高透明度**—您的团队会自然而然地希望在环境中实现更高透明度，从而更轻松地实现主动监控，最终帮助团队成员顺利发现问题。透明度提升能够帮助我们更轻松地找到问题根源，最终延长正常运行时间并改善服务质量。

- **建立更多自动化机制**—开发人员痛恨简单重复的手动任务，因此相较于强迫他们在生产流程当中重复解决问题，大家应当利用自动化机制提高处理效率，最终保证开发人员能够将精力集中在发生问题的根源身上。

- **持续改进运行质量**—持续改进实践、反馈回路以及测试驱动型开发机制，外加自动化与透明度等因素将为我们的内部与外部客户带来更出色的服务质量。

- **更为确切地掌握客户满意度要素**—“谁构建，谁运行”原则使得整个 IT 团队能够更理解客户的需求与实际感受。此类知识不再仅限于产品或者销售团队之内，开发人员同样可以掌握客户反馈指标，从而量化软件内每项功能对于客户的具体价值。

文化为何如此重要

Puppet Labs 发布的《2015 年 DevOps 状态报告》就曾指出，“文化是 DevOps 当中最为重要的实现因素。[5]”这份报告认为，企业的文化与 IT 绩效之间存在着密切关联。良好的 DevOps 实践能够在企业文化中建立高度信任感，而此类实践同时亦成为 Ron Westrum 所提到的“绩效为本”理念的重要组成部分。图四所示为 Westrum 对三类不同企业文化及其特性做出的比较结论。[6][7]

病态权本位思维	官僚主义规则	绩效为本规则
低合作度	中等合作度	高合作度
打击异见者	忽视异见者	培养异见者
推卸责任	削减责任	分担责任
打压建言	容忍建言	鼓励建言
发生问题时寻找替罪羊	发生问题时进行责任划分	发生问题时进行针对性查询
拒绝创新	认为创新会引发问题	不断实现创新

图四：组织文化学（Westrum，2004 年）

图五从 Puppet Labs 报告当中提出了其它一些实践条目，旨在帮助大家建立强大的 DevOps 文化。

1

生产力型文化的特点	DevOps 实践
高度合作	**跨职能团队-** 众多企业建立起跨联通团队，其工作内容涵盖软件交付流程中的每个功能区划（包括商业分析师、开发人员、质量工程师、运维与安全人员等等）。通过这种方式，每位成员都能够分担产品构建、部署与维护等相关职责。
培养异见者	**过者无罪制度-** 不受指责即不再恐惧; 不再恐惧则团队将可更为高效地发现问题并解决问题。错误总会出现，而建立这种过者无罪的制度能够帮助我们更好地从错误中汲取经验。
风险与分担	**分担责任-** 质量、可用性、可靠性与安全将成为每位成员的职责。改善服务质量的一大可行方式，就是确保开发人员需要分担生产环境下代码维护工作的部分职责。这种协作层面的改进源自职能分担带来的风险降低效果：由于更多成员开始关注软件交付流程，意味着流程或者规划中出现的大多数错误能够确切得到规避。自动化机制还能够降低风险，同时选择正确的工具以实现协作。
鼓励建言	**打破交流壁垒-** 除了建立跨职能团队之外，我们亦应考虑让运维团队与开发团队之间实现协同，并将运维工作规划至软件与交付生命周期之内。
发生问题时进行针对性查询	**过者无罪制度-** 对于意外故障的响应方式正是企业文化的一种典型表现。大家应当将注意力集中在导致失败的条件之上，而非指责引发失败的具体个人，只有这样我们才能真正趋近于生产力型文化目标。
不断实现创新	**实验时间-** 为员工提供随意探索新鲜思维的空间，从而带来更理想的开发成果。目前部分企业会每周为工程技术人员提供实验时段。也有部分企业会举办内部黑客日或者小型会议以共同思路与协作进展。通过这种方式，新型功能与产品将不断涌现，亦代表着员工能够在习惯性的重复工作之外帮助企业创造更多额外价值。

图五：如何建立起生产力型文化

注意事项

- **应当** 考虑采用精益敏捷工程实践与 DevOps 文化以实现业务改造。
- **应当** 分析现有服务交付流程以判断哪些要素需要变更，从而支持云 IT 服务的实际采用。
- **应当** 定义 CIO 需要在董事会上进行报告的各关键性指标。
- **不应**在云环境中继续使用内部 IT 服务所惯用的孤立流程与工具。
- **不应**急于设定服务水平协议（简称 SLA），除非您拥有相关举措能够测试其合规性水平。通过比赛日活动，大家可以使用自动化脚本将故障场景引入系统当中，从而测试自身 SLA 合规效果。

投资组合管理

作为 AWS CAF 流程论点中的组成部分，投资组合管理负责为高层管理人员提供对新型投入的管理方法帮助其判断现有投资所交付的预期成果是否良好。如果您目前采取了此类管理方案，则只需要调整投资组践以纳入云 IT 服务即可。如果大家尚未采取此类方案，则可首先建立一套现有资产清单，并立足于服务等管理平台对各新型产品与服务进行优先级排序。

提升规划周期频率

传统投资组合管理通常以年为执行单位，且每季度进行审查并重新规划其优先级顺序。我们应考虑逐步转精益投资组合管理实践，包括将持续规划与管理战略相结合以缩短解决方案的价值交付周期。精益投资组理采用的思维方式类似于高速股票交易流程，其中多个项目同时存在且频繁变动、重新排定优先次序并根具体表现、附加价值、重要性以及相关性重新进行预算分配。要让这类方案发挥效果，首先要摒弃那些造可变动”计划与项目的实践举措。通过采用这一新方案，大家将拥有充分的敏捷性能力，进而根据业务需变做出快速反应。

服务目录属于 IT 规划与交付周期的一大基础性实现前提。其包含众多细节信息，例如 IT 系统所能支持的功能以及系统的采购与托管方式等等。这部分信息可作为投资组合分析与排序工作中的重要输入素材。将的服务目录与治理手段相结合，我们将能够不断提升投资组合管理实践的成熟度水平。

使用模型驱动型与数据驱动型方案以测试特定优先级重估场景的实际效果，即考察其是否能够切实随同业更速度实现投资组合优先级调整。举例而言，使用建模技术以测试延时成本会给优先级结论带来怎样的又将如何左右投资组合规划当中对价值的具体理解方式。

通过投资组合治理指导资源分配

作为 AWS 云采用战略中的组成部分，大家可以推动业务与 IT 部门间规划与管理工作的紧密合作。通过务与技术团队间建立起合作关系，您将能够确保开发团队有机会直接同业务支持方携手共进。这种协作会更高的功能质量水平与更理想的客户满意度。

投资组合管理应当成为您的关键性治理主体，其负责控制哪些客户需求为何又怎样遍历您的 IT 生命周期资组合管理、计划管理与业务运营团队将协同维护业务的正常推进。图六所示即为一套治理模式示例，其述了各部门之间如何协作以加快生产环境下的功能交付速度。

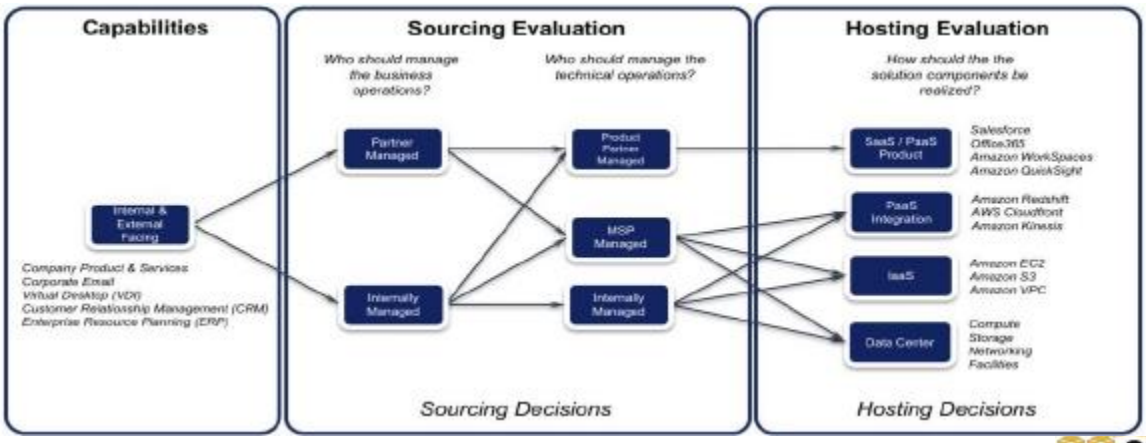
图六：治理模式

在理想情况下，将投资组合管理功能转化为卓越中心（简称 CoE）功能可以为各团队带来更理想的自主
同时改善其响应业务需求变化的能力。

先采购，后托管

很多客户将 AWS 服务视为一套可用于增强其现有工具套件的技术工具组合。他们接触到了 AWS 计算、
与私有网络服务等项目，而后开始将自己的当前应用组件同这些基础性服务进行对应。这种作法虽然有
只会进一步强化以构建者为中心的 IT 功能交付模式。我们发现，那些首先回顾自身业务与 IT 功能采购
企业客户，往往能够更好地利用 AWS 实现业务价值。

图七中的功能采购模式能够指导大家通过一系列决策完成业务与 IT 功能的采购与托管流程。我们建议
扩展自己的现有采购模式，或者据此建立能够满足特殊需求的定制化基础模式。



图七：功能采购模式

我们还建议您利用云至上方案指导采购工作。大家应当优先考量软件即服务（简称 SaaS）选项。如果
不具备可行性，则可考虑平台即服务（PaaS）、基础设施即服务（IaaS）、商业现成方案（COTS）或者
决方案（按此顺序逐一考量）。这套方案能够限定您所需编写及维护的代码量，从而有效控制实现特定
必需的运营成本。

实现可视性与协作性

由于 IT 环境下业务需求与变更发生的速度极快，因此相关管理工具可能需要提供更具分布特性的投资
理方案。具体包括自动实现投资组合监控与变更管理的工具选项。

注意事项

- **应当**将采购策略与托管策略区分开来，从而尽可能在扩展特性与功能的同时降低代码库规模。
- **应当**首先设定一项软件即服务（简称 SaaS）原则，而后分别考虑利用平台即服务（简称 PaaS）与

施即服务（简称 IaaS）以控制支持各项功能所必需的代码规模。

- **应当**在投资组合管理方案中引入一套负责治理的决策制定流程。
- **不应**利用整体式与太过详尽的规划流程处理投资组合管理工作。考虑为开发团队提供一套资源池，

于宏观技术角度审视组合管理事务。

· **不应**将 IT 投资组合管理从业务投资组合规划以及战略制定工作中剥离出来。考虑将 IT 投资组合管理为 CoE 资源，并由 IT 组织与业务部门加以共享。

· **不应**要求投资组合管理团队负责执行整个投资组合管理流程。考虑由投资组合管理团队负责治理、管及报告等与支出相关的工作，而由开发团队负责根据所需成果进行执行流程策划。

计划与项目管理

作为 AWS CAF 流程论点中的组成部分，计划与项目管理提倡在自主权与治理要求之间取得平衡点，从而大家利用云 IT 服务采用举措实现价值提升。项目投资组合管理中的精益方法能够帮助相关团队以自主方成决策制定、决策调整以及业务需求变更管理等工作。在这种情况下，他们将拥有充分的自由空间以调整规模、变更治理边界并选择最合适的方法与工具。这种自主权能够让他们更加主动地交付可切实满足业务的解决方案。

成果驱动型规划

大家应当开发出新的或者调整现有功能，从而高度关注希望达到的既定目标。您可以利用面向目标的工程

技术作为指导。您需要了解产品在每一轮发布时应当拥有怎样的演进效果，从而尽可能缩短审查产品最
可行性的时间周期。这套方案将帮助大家降低前期资金投入与规划压力。

将客户反馈回路引入产品将帮助大家更为明确地制定产品演进决策，并充分立足于实践验证做出判断。
使用 A/B 测试以运行受控实验，同时测试各类构想以思考哪种选项最符合客户对成果的预期。

复杂企业系统的挑战

我们发现，很多客户在对自身大型企业及/或遗留系统加以演进时面临诸多挑战（例如 ERP、CRM 等等
于大型系统重构工作，最为常见的方案之一就是制定一项多年计划以开发出“全新”系统。大家可能在
涯当中经历过一个或者多个此类项目，并深刻理解与之相关的时间、成本以及风险因素。要解决这些挑
大可行方法在于采取“分段应用”而非“一波流”式举措。Martin Fowlers 表示，要使用分段应用方案
家需要“逐步建立一款新型应用，并利用其在相当长的时间周期内替换原有系统，整个周期甚至可能长达
[8] 利用这种方法，大家可以通过一系列小型且易于管理的项目完成对庞大遗留系统的演进。

规模扩展的同时保持团队小型化

Amazon 利用众多小型“双披萨”团队开发并操作我们的服务与新型功能。这些团队只由数位成员组成
位成员身兼至少一种学科门类，包括开发、自动纪、测试、安全、运维乃至产品开发等等。每个团队独
以进行客户反馈收集、发展路线图维护、新功能交付、可靠性保障以及服务的合规与监管要求符合等工
Amazon 的规模化创新工作当中，我们主张利用小型团队避免常见于大规模组织内的沟通障碍问题。

注意事项

- **应当** 考虑利用小规模项目为新型功能开发交付技术方案。利用“分波次推出”计划与项目策划方法
而保证规划内容更具持续性。
- **应当**考虑利用项目成果作为现有解决方案的发展组成部分，而非尝试对整体解决方案进行替换。
- **应当** 定义适合的指标以捕捉并沟通生命周期流程与举措的具体效果。
- **应当**确保利用监控机制确定并解决不同服务举措之间可能存在的潜在冲突。
- **不应**在未进行合规测试之前急于定义 SLA。考虑利用比赛日测试技术向系统中引入故障因素，从而
SLA 合规性水平。
- **不应**试图以单一项目方式直接替换大规模系统及解决方案，考虑将大型项目拆分成多个独立的小型
为其制定更短的执行周期。
- **不应**假定我们必须保留全部现有流程。随着向云环境的转移，流程变更亦将成为一种必然。
- **不应**利用自上而下的方式制定规划。考虑为开发团队提供充足的自由空间，允许他们与业务团队达

性的既定目标实现途径。

持续集成与持续交付（CI/CD）

作为 AWS CAF 流程论战中的绝大部分，CI/CD 旨在帮助企业专注于利用敏捷软件开发实践以实现功能的
式交付。传统的瀑布式开发模式是一种有序型软件开发生命周期，其中各进程环节通过既定阶段如瀑布
推进，具体包括立项、需求、分析、设计、构建、测试、实现与维护等。现代敏捷软件开发机制则更多
量化开发思路，包括以迭代方式实现软件的规划、设计、开发与测试。

利用敏捷模式，大家可以通过 CI/CD 实践及相关工具以自动化方式完成软件的交付生命周期，包括其
动化构建、部署与测试工作。这种作法能够帮助我们实现产品的快速交付与持续改进。

实现持续集成

要实现 CI 目标，我们需要将软件当中提交的任何一项变更纳入到项目的版本控制库当中。考虑使用基
的开发方式以达成这一目标。在基于主干的开发方案当中，每位成员提交的变更都会在当天被纳入主干
每次纳入都会触发一项构建与自动化测试操作。当测试发现需要进行版本回滚时，开发人员或者团队负
当前功能的开发，而后重新回归上一正常版本。如果他们无法在数分钟内完成回滚，则必须保留这项变
如果全部开发团队皆遵循这一方法，那么 CI 也将就此实现。

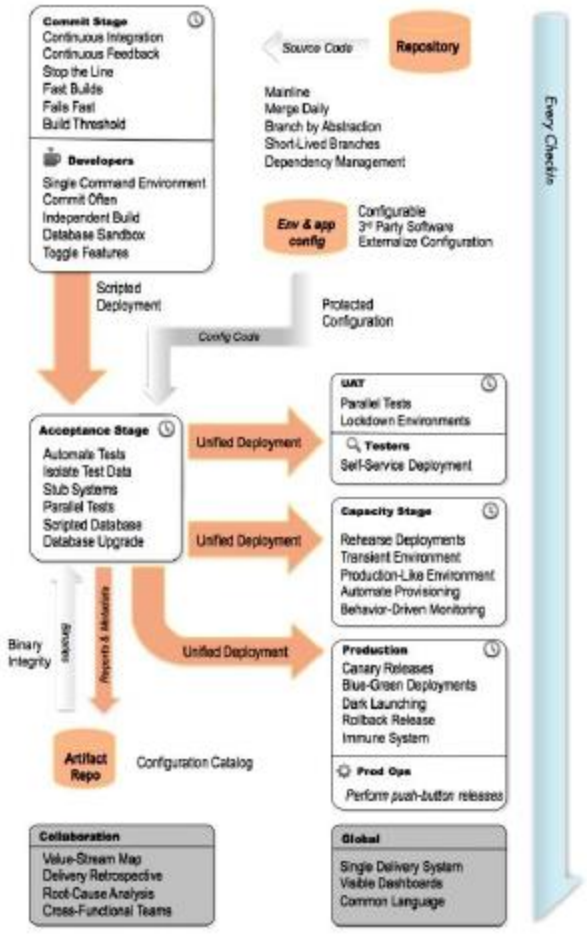
实现部署通道

一部分专家将持续交付（简称 CD）的核心定义为“如何让多个活动组件加以结合，具体包括配置管理、自动化测试、持续集成与部署、数据管理、环境管理以及发布管理等。”[10] 这种配置模式将允许环境本身支持团队的交付能力，并帮助其在 CI/CD 环境下积累经验。Paul Duvall[11] 为这种部署通道总结出三项目的：

· 可视性：交付系统中的各个层面——包括构建、部署、测试与发布——皆可为协作体系中各团队中的每位成员所查看。

· 反馈：团队成员能够在最短时间内发现问题，同时在最短时间内解决问题。

· 持续部署：通过一套完全自动化流程，大家可以立足于任意环境对任意软件版本进行部署与发布。



图八：交付通道的各组成部分（Duvall） [8]

部署并非发布

首先需要对技术决策（例如部署某款产品以及制定一项业务决策）与发布产品进行明确区分。Jez Humble 和
发布是指“将一项功能或者一套功能组合向客户交付的过程。”软件发布属于一项商业决策。他将部署定义
特定软件版本安装至特定环境当中。”产品的实际部署属于一种技术决策。[12]

系统层级部署

考虑使用蓝绿部署，这是一项持续交付技术，能够缩短由发布到生产之间的交接周期。通过这种方法，我
以同时保有两套尽可能接近一致的生产环境。在任何时间点上，这两套生产环境中的一套（假设为蓝）处
动状态。而在筹备下一轮发布时，您将会利用其中的另一套环境（假设为绿）进行具体测试。在测试完
我们将把生产流量由绿重新定向至绿。这时蓝将处于闲置状态，直到下一轮发布就绪才会接手最终测试

功能层级部署

静默启动属于一项持续交付技术，大家可以用于发布一种到多种功能或组件。开发人员可以利用功能标签
制哪位客户能够访问哪项功能。此技术支持 A/B 测试，同时亦可减缓客户功能数量的上升速度。

注意事项

- **应当**考虑过渡至开发/安全/运维类开发环境，其中开发团队内部同时拥有开发、安全与运维技能储
- **应当** 通过使用 A/B 测试技术以验证各类构想，从而充分发挥直接客户数据的优势。
- **应当**创建一套强大的流程以管理每天数以千计的各类变更。
- **不应**在设置 AWS 环境时采用不包含 CI/CD 的实现模式。即使大家暂时不需要转移至 CI/CD 环境，

支持能力也是必不可少的。

流程自动化

作为 AWS CAF 流程论点中的组成部分，流程自动化能够帮助企业在由云采用向更高层级迈进时获得可观
值回报。云技术的加入消除了大部分与传统 IT 管理工作相关的简单重复劳动，而自动化则能够帮助企业
多 IT 领域提高效率并增强业务关注度。

大家已经可以利用多种成熟工具实现流程自动化——其中包括 AWS CloudFormation、Chef、Puppet 以
Ansible 等等。

您应当在面对以下问题时考虑选择流程自动化方案作为解决办法：

- “我该如何利用云技术节约资金？”
- “我该如何保证应用程序安全性？”
- “我该如何避免被锁定在单一平台之上？”
- “我该如何加快推进速度？”

Stephen Orban 对于自动化的细节助益做出了探讨，他给出的结论包括：:13

效率—作为高层管理人员，大家的主要职责之一在于专注于提升各类倡议当中的资源比例，从而帮助业务
更为可观的营收。这亦成为使用云服务的一大关键动力——通过削减投入至基础设施管理领域的支出，我
够将这部分资源转而用于产品开发。而通过以自动化方式处理各类重复任务，具体包括为操作系统安装
应用程序部署、防火墙变更以及监控/警报配置等，您将能够将更多时间投入到功能开发当中。

弹性—这是云平台最大的优势之一。凭借着自动化系统，大家能够通过脚本反复使用各类 AWS 服务—

Auto Scaling——从而根据实际需求进行容量调整。容量规划一般难于实现，而在这方面犯下的错误有可终导致巨大的资源浪费以及/或者糟糕的客户使用体验。

移植性—当我们能够以可靠方式将某一系统中的架构或者基础设施重现在另一系统之上时，则意味着发现管方式需要变更时，大家可以更轻松地变更现有自动化脚本——而不必单纯以手动方式完成平台定制。

安全性—以自动化方式将良好安全实践引入系统，能够使得我们的系统更轻松地完成均匀扩展、消除人为导致的数据泄露可能性，同时允许大家更为快速且自信地推出安全改进（与推出其它功能一样）。一部分色的架构系统则会以分段方式运用自动化机制。在每个阶段中，此类系统都会执行某项特定操作——例如某套操作系统、从内部库中下载代码、将应用程序上传至互联网，同时只允许保留每阶段所必需的连接。种作法能够确保来自互联网的流量不会接入企业内部系统。

审计性—利用自动化脚本构建的系统在运作层面往往更具可预测性，且拥有更多记录信息。任何自动执行作都会依实际顺序被记录在案。这就使我们能够更轻松地将其状态向他人传达——包括团队成员、业务相士、审计人员以及监管人员等等，帮助他们了解谁在系统中何时执行过哪些操作。

可恢复性—在恢复方面，构建自动化系统至少能够带来两大核心助益。首先，随着大家在常规自动化操作中不断积累经验，正常的自动运行状态检查以及常规质量检查会变得更为简单。自动化机制还能够被用于回滚那些无法通过这些检查的系统。第二，由于环境变更难度不断下降，大家将能够更为激进地推出某些功并在其无法获得预期效果时随时回滚或者在此基础上安装后续补丁。相比之下，这种能力在过去需要耗费以手动完成质量检查的背景下是根本无法想象的。

采用 AWS 服务能够帮助您利用等同于软件变更管理的方式掌控基础设施变更。根据您需要的基础设施环建相应定义，而后将其作为软件代码加以维护与存储即可。

注意事项

- **应当** 在代码库当中更新并存储各项基础设施定义。
- **应当** 创建一套自动化服务目录。
- **应当** 预先创建规模伸缩策略，用于指导云采用实践。
- **应当**预先创建一套多租户配置策略，用于指导云采用实践。
- **不应** 坐等失败状况出现后才做出反应。考虑预先建立一套灾难恢复策略，并用于指导云采用实践。

质量管理

作为 AWS CAF 的组成部分，质量管理鼓励使用精益与敏捷原则及实践，旨在帮助企业交付高质量产品
客户满意度并实现更具可持续性的竞争优势。遵循这些原则将帮助您所在企业的员工实现持续改进，将
求融入文化与企业活动。而采用基于云的服务以交付软件产品，正是实现上述成果的关键性前提之一。

精益原则

丰田公司于第二次世界大战之后发展出了精益生产理论，并借此长成为全球规模最大的汽车制造商，同
质与创新领域取得高度评价。在过去二十年中，越来越多制造业之外的企业开始采用精益原则以解决类
务挑战。这项理论中的五大首要原则 [14] 包括：

1. **确定客户与指定价值**—其出发点在于，意识到任何企业都仅有一小部分时间与精力投入能够切实转
终客户的附加价值。为了立足于最终客户角度明确定义特定产品或者服务的价值所在，一切非价值举措
者浪费行为——都应有针对性地进行去除。

2. **确定并映射价值流程**—价值流程是指企业在进行产品或者服务交付的过程当中，涉及的全部相关举
代表着向客户交付价值的整个端到端流程。在理解了客户下一阶段拥有怎样的实际需求之后，我们将能
清晰地设计具体交付方式（或者回避错误的交付方式）。

3. **通过杜绝浪费建立流程**—在初步进行价值流程映射时，我们往往会发现只有 5%的举措能够带来附加
这一比例可能在服务环境当中提升至 45%。杜绝这种浪费，我们可以确保自己的产品或者服务“流至”
且其间不存在任何中断、迂回或者等待状况。

4. **响应客户要求**—理解客户对于服务的需求，而后创建对应流程对响应此种需求。其目标在于只在客
时据此建立生产方式。

5. **追求完美**—创建流程并从根本层面重构各流程步骤，但实际结果仍然取决于各步骤间的整合效果。随
的演进，我们会发现越来越多浪费状况，而流程也将不断向理论层面的完美方向推进。从完美的角度理解
资产与每项操作都应能够为最终客户提供附加价值。

通过遵循以上五项精益原则，大家将树立一种坚定的实现原则，即“一切为了目标服务”。其将帮助大家
己不断检讨整体组织战略流程，同时保证其始终为客户提供一致的附加价值。这将使得您的企业在保持较
的同时，不断根据业务需求推动服务发展，并通过可持续性变革实现这项目标。

敏捷

敏捷是指于 2001 年建立的《敏捷宣言》中定义的各项价值与原则。这份宣言反映了软件开发方法（例如
开发方法）的地位正逐步提升，且各位宣言作者亦意识到现有方法在根据客户需求交付软件方面存在严重

《敏捷宣言》当中强调的价值取向包括流程与工具的个别与交互作用、通过全面文档实现的可工作软件、
谈判实现的客户协作以及以规划遵循实现的变更响应等等。

敏捷概念包含以下几项原则：

- 最高优先级要求为客户满意度
- 对要求变更持欢迎态度
- 高频度软件交付
- 业务人员与开发人员间的日常合作
- 围绕动机进行项目创建
- 最好选择面对面交流方式
- 利用工作软件衡量当前进度
- 以可持续发展节奏进行项目推进
- 持续关注技术卓越性
- 重视简化
- 自组织团队
- 定期反思与调整

作为宣言签名者之一，Jeff Sutherland 指出，敏捷开发本身并不是一种具体方法，而更像是一种用于描述
捷方法的统称。此类敏捷方法具体包括 [16]：

- Scrum 或者 Kanban 管理方法
- 极限编程
- 持续集成
- 持续交付
- 功能驱动型开发
- 测试驱动型开发

每项单独敏捷方法在价值实现方式层面都有着相当程度的共性，但这些方法亦全部拥有特殊的流程与实践
此实现一种或者多种价值回报。软件团队可以利用这些价值与原则，同时配合敏捷开发方法以实现真正的
成效。

注意事项

- **应当** 采用精益原则与学习套路在企业当中提高生产力并减少浪费现象，同时改善客户满意度。
- **应当** 采用敏捷价值、原则与方法进行软件交付，从而满足客户在功能、速度以及质量方面的需求。
- **应当** 利用价值流程图以实现系统持续改进。
- **不应**错误地认为通过挑选部分敏捷实践即可实现真正敏捷采用所带来的成果。

总结

意识到企业业务流程变革迫切性是充分发挥云 IT 服务优越性与价值的首要前提。下一步则在于选定具体
您可以选择稍做改变以确保云系统拥有可持续运营能力，并通过关注 IT 运营逐步实现附加价值。而作为
极端，您也可以选择更为广泛的改造方式，从而使得业务迎来颠覆性的变化与发展趋势。AWF CAF 流程
够帮助大家审视既有方法选项，并考虑哪种方法最适合您的实际需要。

CAF 分类与术语

云采用框架（简称 CAF）是一套由 AWS 创建的框架，用于从以往客户的参与活动中收集指导依据与最佳实
一项 CAF“论点”代表的是与云 IT 系统在企业内实现所需要高度关注的某一相关领域。举例而言，成熟
点主要着眼于企业人员、流程、工具以及技术的成熟程度，并负责提供指导意见以检测您的企业是否已经
了由现有状态转为云业务运作的准备。

每项 CAF 论点都由多项组成部分及举措构成。其中，每项组成部分代表的是论点的一种子领域，指代企
要关注的某个特定方向。本份白皮书探讨了成熟度论点范畴之内的各组成部分。另外，举措代表的则是实
行性规划的指导性意见，企业可利用其完成云环境迁移并操作各类云解决方案。

举例来说，服务交付管理即是流程观点中的一项组成部分，而其中采用精益-敏捷工程实践的具体作法则
种实际举措。

而通过进一步合并，云采用框架（简称 CAF）与云采用方法（简称 CAM）二者可作为指导方针，帮助企
顺利地完成 AWS 云过渡之旅。

备注：

[1] https://d0.awsstatic.com/whitepapers/aws_cloud_adoption_framework.pdf

[2] Edwards, Damon，2010 年。《DevOps 并非技术问题，而是一项业务问题。-dev2ops-dev2ops》检索于
年 9 月 7 日
(http://dev2ops.org/2010/11/devops-is-not-a-technology-problem-devops-is-a-business-problem/)。

[3]
2015 与 Werner Vogels 间的对话——ACM 队列，检索于 2015 年 9 月 21 日
(http://queue.acm.org/detail.cfm?id=1142065)。

[4] Orban, Stephen，2015 年。《企业 DevOps：我们为何应当遵循谁构建、谁运行原则——AWS 企业系列—
检索于 2015 年 9 月 9 日
(https://medium.com/aws-enterprise-collection/part-3-in-the-enterprise-devops-series-why-you-should-run-what-you-build-c62f0990f4c3)。

[5]
abs、Puppet 与 IT Revolution 出版社。2015 年《2015 年 DevOps 报告》（877 页）。
(https://puppetlabs.com/sites/default/files/2015-state-of-devops-report.pdf)。

[6] Westrum, R，2004 年。《组织文化类型学》，检索于 2015 年 11 月 10 日
(http://qualitysafety.bmj.com/content/13/suppl_2/ii22.full.pdf+html)。

[7] Humble、 Jez、Joanne Molesky 与 Barry O'Reilly，2014 年。《精益企业：高绩效组织如何在规模化环
现创新》，检索于 2015 年 5 月 17 日
(http://www.amazon.com/gp/product/1449368425)。

Fowler, Martin，2004 年。《吞噬型应用》，检索于 2015 年 5 月 17 日 (http://www.martinfowler.com/bliki/StranglerApplication.html)。

Duvall, Paul，2011 年。《持续集成——Dzone——记录》，检索于 2015 年 9 月 6 日 (https://dzone.com/refcardz/continuous-integration)。

[10] Jez Humble 与 David Farley，《持续交付：通过自动化构建、测试与部署实现可靠软件发布》，Addison W 教授，2010 年。

[11] Duvall, Paul. 2011. “Continuous Delivery - DZone - Refcardz.” Retrieved September 6, 2015 ，2011 年。《持续交付——DZone——记录》，检索于 2015 年 9 月 6 日 (https://dzone.com/refcardz/continuous-delivery-patterns)。

[12] Jez Humble 与 David Farley，《持续交付：通过自动化构建、测试与部署实现可靠软件发布》，Addison W 教授，2010 年。

[13]

Orban, Stephen，2015 年。《让自动化成为云战略中的关键性组成部分——AWS 企业系列——中》，检索于 年 9 月 9 日 (https://medium.com/aws-enterprise-collection/make-automation-a-key-part-of-your-cloud-strategy-8eea07b0986c)。

[14] 精益大学。《精益思维中的五项基本原则》 (http://www.cardiff.ac.uk/lean/principles/)。

[15] http://agilemanifesto.org/history.html

[16] Sutherland, Jeff，《敏捷原则与价值》 (https://msdn.microsoft.com/en-us/library/dd997578(v=vs.120).aspx)。