



针对 Amazon EKS 应用程序的高可用性和弹性进行设计

# AWS 规范性指导



# AWS 规范性指导: 针对 Amazon EKS 应用程序的高可用性和弹性进行设计

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

# Table of Contents

简介 .....	1
高可用性和弹性设计 .....	2
分散工作负载 .....	2
使用 pod 拓扑分布约束 .....	2
Pod 亲和力和反亲和力 .....	6
Pod 中断预算 .....	8
探测器和健康检查 .....	8
启动探测 .....	9
活性探测器 .....	9
准备情况调查 .....	9
入口资源和负载均衡器运行状况检查 .....	10
容器生命周期挂钩 .....	10
了解区域中断期间的 pod 驱逐情况 .....	12
实施 Amazon EKS 区域转移以提高弹性 .....	12
了解区域偏移机制 .....	12
区域移位激活方法 .....	13
有效区域偏移的先决条件 .....	13
关于区域干扰弹性的建议 .....	14
轮班完成和恢复 .....	14
结论 .....	15
Resources ( 资源 ) .....	16
文档历史记录 .....	17
术语表 .....	18
# .....	18
A .....	18
B .....	21
C .....	22
D .....	25
E .....	28
F .....	30
G .....	31
H .....	32
我 .....	33
L .....	35

M .....	36
O .....	40
P .....	42
Q .....	44
R .....	45
S .....	47
T .....	50
U .....	51
V .....	52
W .....	52
Z .....	53
	liv

# 专为 Amazon EKS 应用程序的高可用性和弹性而设计

Amazon Web Services 的冯浩飞、Frank Fan 和 Rus Kalakutskiy ()AWS

2025 年 10 月 ( [文档历史记录](#) )

确保应用程序设计中的高可用性 (HA) 和弹性对于实现接近零的恢复点目标 (RPO) 和恢复时间目标 (RTO) 至关重要。随着越来越多的组织将其应用程序迁移到 Kubernetes 环境并对其进行现代化改造，对强大且可扩展的解决方案的需求不断增加。Amazon Elastic Kubernetes Service ( Amazon EKS ) 可帮助您高效地大规模管理容器化应用程序。

本指南深入探讨了一系列广泛认可的设计和管理 Amazon EKS 微服务应用程序的建议和最佳实践。基于丰富的经验和实际部署情况，这些见解为架构师和开发人员提供了宝贵的指导。实施这些建议，提高基于 Kubernetes 的应用程序的性能、可靠性和可扩展性，从而实现稳健的操作。

# 高可用性和弹性设计注意事项

Kubernetes 的分担责任模型变得更加复杂。Amazon EKS 控制平面的可用性和弹性由亚马逊 Web Services (AWS) 管理。您的组织管理数据平面，这可能会严重影响微服务应用程序的性能和可用性。

在 Amazon EKS 上设计高可用性和弹性的应用程序时，请考虑以下组件：

- 微服务应用程序：其 pod 和容器
- 工作负载数据平面：入口控制器、pod、系统组件，例如[亚马逊虚拟私有云 \(Amazon VPC\) 容器网络接口 \(CNI\)](#)、服务网格 sidecar 和 kube-proxy
- 工作负载管理层：控制器、准入控制器、网络策略引擎以及这些组件的持久数据存储
- Kubernetes 控制平面
- 基础架构：节点、网络和网络设备

对于前三个注意事项（指在 Kubernetes 集群中运行的组件），本指南涵盖以下主题：

- [在节点和可用区之间分配工作负载](#)
- [使用 PDB 保护关键工作负载](#)
- [配置探测器和运行状况检查](#)
- [配置容器生命周期挂钩](#)
- [了解区域中断期间的 pod 驱逐情况](#)

## 将工作负载分散到节点和可用区中

在可用区和节点等[故障域](#)之间分配工作负载可提高组件可用性并降低水平可扩展应用程序的故障几率。以下各节介绍了在节点和可用区之间分配工作负载的方法。

### 使用 pod 拓扑分布约束

[Kubernetes 容器拓扑分布约束](#)指示 Kubernetes 调度器在不同的故障域（可用区、节点和硬件类型）之间分配由ReplicaSetStatefulSet不同故障域（可用区、节点和硬件类型）管理的 Pod。使用 pod 拓扑分布约束时，可以执行以下操作：

- 根据应用程序要求将容器分布或集中到不同的故障域中。例如，您可以分发 Pod 以提高弹性，也可以将 pod 集中在网络性能上。

- 组合不同的条件，例如跨可用区分发和跨节点分配。
- 如果无法满足条件，请指定首选操作：
  - `whenUnsatisfiable: DoNotSchedule`结合使用`maxSkew`和`minDomains`来为调度程序创建硬性要求。
  - `whenUnsatisfiable: ScheduleAnyway`用于减少`maxSkew`。

如果故障区域变得不可用，则该区域中的 Pod 将变得不健康。Kubernetes 会重新调度 Pod，同时尽可能遵守传播限制。

以下代码显示了跨可用区或跨节点使用 pod 拓扑分布约束的示例：

```
...
spec:
  selector:
    matchLabels:
      app: <your-app-label>
  replicas: 3
  template:
    metadata:
      labels: <your-app-label>
    spec:
      serviceAccountName: <ServiceAccountName>
  ...
    topologySpreadConstraints:
      - labelSelector:
          matchLabels:
            app: <your-app-label>
        maxSkew: 1
        topologyKey: topology.kubernetes.io/zone # <---spread those pods evenly over
all availability zones
        whenUnsatisfiable: ScheduleAnyway
      - labelSelector:
          matchLabels:
            app: <your-app-label>
        maxSkew: 1
        topologyKey: kubernetes.io/hostname # <---spread those pods evenly over all
nodes
        whenUnsatisfiable: ScheduleAnyway
```

## 默认集群范围的拓扑分布限制

默认情况下，Kubernetes 提供了一组拓扑分布限制，用于跨节点和可用区分配 Pod：

```
defaultConstraints:
- maxSkew: 3
  topologyKey: "kubernetes.io/hostname"
  whenUnsatisfiable: ScheduleAnyway
- maxSkew: 5
  topologyKey: "topology.kubernetes.io/zone"
  whenUnsatisfiable: ScheduleAnyway
```

### Note

需要不同类型拓扑约束的应用程序可以覆盖集群级别的策略。

默认约束设置得很高maxSkew，这对于拥有少量 pod 的部署没有用。到目前为止，KubeSchedulerConfiguration[无法在 Amazon EKS 中进行更改](#)。如果您需要强制执行其他组拓扑分布约束，请考虑使用变更准入控制器，如下节所示。如果您运行备用调度程序，也可以控制默认的拓扑分布约束。但是，管理自定义调度程序会增加复杂性，并可能对集群弹性和高可用性产生影响。出于这些原因，我们不建议仅对拓扑分布约束使用其他调度器。

## 拓扑分布限制的网守策略

强制执行拓扑分布限制的另一种选择是使用[Gatekeeper](#)项目中的策略。网守策略是在应用程序级别定义的。

以下代码示例演示如何使用Gatekeeper OPA策略进行部署。您可以根据需要修改策略。例如，仅将策略应用于带有标签的部署HA=true，或者使用不同的策略控制器编写类似的策略。

第一个示例显示了与以下ConstraintTemplate内容一起使用k8stopologyspreadrequired\_template.yml：

```
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8stopologyspreadrequired
spec:
  crd:
```

```
spec:
  names:
    kind: K8sTopologySpreadRequired
  validation:
    openAPIV3Schema:
      type: object
      properties:
        message:
          type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8stopologyspreadrequired

        get_message(parameters, _default) =3D msg {
          not parameters.message
          msg :=_default
        }

        get_message(parameters, _default) =3D msg {
          msg := parameters.message
        }

        violation[{"msg": msg}] {
          input.review.kind.kind ="Deployment"
          not input.review.object.spec.template.spec.topologySpreadConstraint
          def_msg :"Pod Topology Spread Constraints are required for Deployments"
          msg :get_message(input.parameters, def_msg)
        }
}
```

以下代码显示了 constraints YAML 清单 `k8stopologyspreadrequired_constraint.yml`：

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sTopologySpreadRequired
metadata:
  name: require-topologyspread-for-deployments
spec:
  match:
    kinds:
      - apiGroups: ["apps"]
        kinds: ["Deployment"]
```

```
namespaces: ## Without these two lines will apply to the whole cluster
- "example"
```

## 何时使用拓扑分布约束

考虑在以下场景中使用拓扑分布约束：

- 任何水平可扩展的应用程序（例如，无状态 Web 服务）
- 具有主动-主动或主动-被动副本的应用程序（例如，NoSQL 数据库或缓存）
- 带有备用副本的应用程序（例如控制器）

例如，可用于水平可扩展场景的系统组件包括以下组件：

- [集群自动扩缩器和 Karpenter](#)（使用 `replicaCount > 1` 和 `leader-elect = true`）
- [AWS Load Balancer 控制器](#)
- [CoreDNS](#)

## Pod 亲和力和反亲和力

在某些情况下，确保一个节点上运行的特定类型的 pod 不超过一个是有益的。例如，为了避免在同一个节点上调度多个网络密集型的 Pod，您可以使用带有或标签的反关联性规则。Ingress Network-heavy 使用时 `anti-affinity`，您还可以使用以下各项的组合：

- 网络优化节点上的污点
- 对网络密集型 pod 的相应容忍度
- 节点亲和性或节点选择器，用于确保网络密集型的 Pod 使用网络优化的实例

以网络密集型的 pod 为例。您可能有不同的要求，例如 GPU、内存或本地存储。有关其他用法示例和配置选项，请参阅 [Kubernetes 文档](#)。

## 重新平衡吊舱

本节讨论在 Kubernetes 集群中重新平衡 Pod 的两种方法。第一个使用了 Kubernetes 的解调度器。Descheduler 通过强制执行策略来移除违反拓扑分布限制或反关联性规则的 pod，从而帮助维持 pod 分布。第二种方法使用 Karpenter 合并和装箱功能。整合通过将工作负载整合到更少、打包效率更高的节点上，持续评估和优化资源使用情况。

如果你不使用 Karpenter，我们建议你使用 Descheduler。如果您同时使用 Karpenter 和 Cluster Autoscaler，则可以将 Descheduler 与 Cluster Autoscaler 一起用于节点组。

## 无组节点的解调器

移除 pod 后，无法保证拓扑约束仍然得到满足。例如，缩小部署规模可能会导致 Pod 分布不平衡。但是，由于 Kubernetes 仅在调度阶段使用 pod 拓扑分布约束，因此 Pod 在故障域中处于不平衡状态。

为了在这种情况下保持平衡的容器分布，你可以使用适用于 Kubernetes 的 [Descheduler](#)。Descheduler 是一个有用的工具，可用于多种用途，例如强制执行 pod 的最大使用寿命或存活时间 (TTL)，或者改善基础设施的使用。在弹性和高可用性 (HA) 的背景下，考虑以下 Descheduler 策略：

- [RemovePodsViolatingTopologySpreadConstraint](#)
- [RemovePodsViolatingInterPodAntiAffinity](#)
- [RemoveDuplicates](#)

## Karpenter 整合和装箱功能

对于使用 Karpenter 的工作负载，您可以使用整合和垃圾箱打包功能来优化资源利用率并降低 Kubernetes 集群的成本。Karpenter 会持续评估 pod 的放置位置和节点利用率，并尝试尽可能将工作负载整合到更少、打包效率更高的节点上。此过程包括分析资源需求，考虑 Pod 亲和性规则等限制，以及可能在节点之间移动 Pod 以提高集群的整体效率。以下代码提供了一个示例：

```
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: default
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
    expireAfter: 720h
```

对于consolidationPolicy，你可以使用WhenUnderutilized或WhenEmpty：

- 当设置consolidationPolicy为WhenUnderutilized，Karpenter 会考虑所有节点进行合并。当 Karpenter 发现一个空的节点或未充分利用时，Karpenter 会尝试移除或替换该节点以降低成本。

- 当设置 `consolidationPolicy` 为时 `WhenEmpty`，Karpenter 只考虑不包含工作负载 Pod 的节点进行整合。

Karpenter 整合决策不仅仅基于您在监控工具中可能看到的 CPU 或内存利用率百分比。相反，Karpenter 使用更复杂的算法，该算法基于 pod 资源请求和潜在的成本优化。有关更多信息，请参阅 [Karpenter 文档](#)。

## 使用 PDB 保护关键工作负载

Pod 中断预算 (PDB) 是维护集群中应用程序高可用性的基本功能。PDB 指定目标大小，即特定类型的 pod 的最低可用性。这意味着，在任何给定时间都必须有最少量的特定 pod 类型的副本在运行。如果正在运行的副本数量低于目标大小，Kubernetes 将防止对剩余副本的进一步中断，直到达到目标大小。PDBs 有助于确保工作负载不受这些事件的影响，并且可以继续不间断地运行。发生中断时，Kubernetes 会尝试优雅地将 Pod 从受影响的节点中移出，同时保持 PDB 中指定的副本数量。

您可以使用 PDB 来声明副本的 `maxUnavailable` 数量 `minAvailable` 和数量。例如，如果您希望应用程序至少有三个副本可用，请创建一个类似于以下示例的 PDB：

```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: my-svc-pdb
spec:
  minAvailable: 3
  selector:
    matchLabels:
      app: my-svc
```

PDBs 正确设置应用程序有助于最大限度地减少计划内或计划外事件期间的中断。您可以使用反关联性规则将部署的 pod 调度到不同的节点上，避免节点升级期间的 PDB 延迟。

## 配置探测器和负载均衡器运行状况检查

除了负载均衡器运行状况检查外，Kubernetes 还提供了多种执行应用程序运行状况检查的方法。你可以将以下 Kubernetes 内置探测器与负载均衡器运行状况检查一起运行，作为容器上下文中的命令或者作为对 kubelet 或主机 IP 地址的 HTTP/TCP 探测器。

存活探测器和就绪探测器应该不同且独立（或者至少具有不同的超时值）。如果应用程序存在临时问题，则就绪探测器会将该 pod 标记为未就绪，直到问题得到解决。如果 liveness 探测器设置不正确，则存活探测器可能会终止 pod。

## 启动探测

使用启动探测器来保护初始化周期较长的应用程序。在启动探测成功之前，其他探测器将处于禁用状态。

你可以定义 Kubernetes 等待应用程序启动的最长时间。如果在配置的最长时间之后，Pod 仍然无法通过启动探测，则应用程序将被终止，并创建一个新的 Pod。

当应用程序的启动时间不可预测时，请使用启动探针。如果您知道应用程序需要 10 秒钟才能启动，请将 initialDelaySeconds 改用存活探测器或就绪探测器。

## 活性探测器

使用活动探测器来检测应用程序问题或进程是否正常运行。活跃探测器可以检测到死锁情况，即进程继续运行但应用程序无响应。使用活性探测器时，请执行以下操作：

- initialDelaySeconds 用于延迟第一次探测。
- 不要为存活和就绪探测器设置相同的规格。
- 不要将存活探测器配置为依赖于 pod 外部因素（例如数据库）。
- 将活性探测器设置为特定 terminationGracePeriodSeconds。有关更多信息，请参阅 [Kubernetes 文档](#)。

## 准备情况调查

使用就绪探测器检测以下内容：

- 应用程序是否已准备好接受流量
- 部分可用性，即应用程序可能暂时不可用，但预计在某个操作完成后会恢复正常

就绪探测器有助于确保应用程序配置和依赖关系在没有问题或错误的情况下运行，以便应用程序可以为流量提供服务。但是，配置不当的就绪探测器可能会导致中断，而不是阻止中断。依赖于外部因素（例如与数据库的连接）的就绪探测可能会导致所有 Pod 都无法通过探测。此类故障可能导致中断，并可能导致从后端服务到使用故障 pod 的其他服务发生级联故障。

## 入口资源和负载均衡器运行状况检查

Application Load Balancer 和 Kubernetes ingress 提供运行状况检查功能。对于 Application Load Balancer 运行状况检查，请指定目标端口和路径。

### Note

对于 Kubernetes 来说 ingress，注销会有延迟。Application Load Balancer 的默认值为 300 秒。考虑使用与准备探测相同的值来设置入口资源或负载均衡器运行状况检查。

NGINX 还提供运行状况检查。有关更多信息，请参阅 [NGINX 文档](#)。

Istio 的入口和出口网关没有与 NGINX 的 HTTP 运行状况检查相提并论的运行状况检查机制。但是，您可以使用 [Istio 断路器或 DestinationRule 异常值检测](#) 来实现类似的功能。

有关更多信息，请参阅 Amazon EKS 最佳实践指南中的 [可用性和 Pod 生命周期](#)。

## 配置容器生命周期挂钩

在容器正常关闭期间，您的应用程序应通过开始关闭来响应 SIGTERM 信号，这样客户端就不会遇到任何停机时间。您的应用程序应运行清理程序，如下所示：

- 保存数据
- 关闭文件描述符
- 关闭数据库连接
- 优雅地完成机上请求
- 及时退出以满足 pod 终止请求

设置足够长的宽限期以完成清理。要了解如何响应 SIGTERM 信号，请参阅用于应用程序的编程语言的文档。

[容器生命周期挂钩](#) 使容器能够感知其管理生命周期中的事件。当执行相应的生命周期挂钩时，容器可以运行在处理程序中实现的代码。容器生命周期挂钩为 Kubernetes 和云的异步性质提供了一种解决方法。这种方法可以防止在入口资源之前转发到终止的 pod 并 iptables 更新为不向 pod 发送新流量的连接丢失。

容器生命周期Endpoint、和EndpointSlice是不同的一部分 APIs。精心策划这些 APIs 很重要。但是，当 Pod 被终止时，Kubernetes API 会同时通知 kubelet（用于容器生命周期）和控制器。EndpointSlice 有关更多信息（包括图表），请参阅 Amazon EKS 最佳实践指南中的[优雅处理客户请求](#)。

当 kubelet 发送 SIGTERM 到 pod 时，EndpointSlice 控制器正在终止 EndpointSlice 对象。该终止会通知 Kubernetes API 服务器通知每个要更新的 kube-proxy 节点。iptables 尽管这些操作是同时发生的，但它们之间没有依赖关系或顺序。容器收到 SIGKILL 信号的可能性很可能比每个节点 kube-proxy 上更新本地 iptables 规则的时间要早得多。在这种情况下，可能的情况包括以下几种：

- 如果您的应用程序在收到请求和连接后立即直言不讳地丢弃了正在进行的请求和连接 SIGTERM，则客户端会看到错误。500
- 如果您的应用程序确保在收到所有进行中的请求和连接后都得到完全处理 SIGTERM，那么在宽限期 内，新的客户端请求仍会发送到应用程序容器，因为 iptables 规则可能尚未更新。在清理过程关闭容器上的服务器套接字之前，这些新请求将产生新的连接。宽限期结束后，发送后建立的新连接将 SIGTERM 无条件断开。

要解决之前的情况，您可以实现应用内集成或 PreStop 生命周期挂钩。有关更多信息（包括图表），请参阅 Amazon EKS 最佳实践指南中的[优雅关闭应用程序](#)。

#### Note

无论应用程序是否正常关闭，或者 preStop 挂钩的结果如何，应用程序容器最终都会在宽限期 结束时终止。SIGKILL

使用带有 sleep 命令的 preStop 挂钩来延迟发送 SIGTERM。这将有助于在入口对象将新连接路由到 pod 时继续接受这些连接。测试 sleep 命令的时间值，确保将 Kubernetes 的任何延迟和其他应用程序 依赖关系都考虑在内，如以下示例所示：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      lifecycle:
```

```
# This "sleep" preStop hook delays the Pod shutdown until
# after the Ingress Controller removes the matching Endpoint or EndpointSlice
preStop:
  exec:
    command:
      - /bin/sleep
      - "20"
    # This period should be turned to Ingress/Service Mesh update latency
```

有关更多信息，请参阅 Kubernetes 文档中的[容器挂钩](#)和 Amazon EKS [最佳实践指南中的优雅关闭](#)应用程序。

## 了解区域中断期间的 pod 驱逐情况

当可用区完全中断时（也就是说，当该可用区中的所有节点都失去与 Kubernetes 控制平面的连接时），Kubernetes 中的[节点生命周期控制器](#)会检测到这种情况并将 Pod 逐出受影响的区域。无法访问的节点上的 Pod 被标记为 `Terminating` 而新的 Pod 则被调度到可用区中运行良好的节点上。在此期间，受影响的节点会显示 `NotReady` 状态，调度器会阻止在这些节点上放置新的 pod，`EndpointSlice` 控制器会从服务路由中删除与受损可用区关联的端点，直到连接恢复。

对于涉及区域内部分节点故障的场景（只有部分节点无法访问），节点生命周期控制器会应用不同的驱逐行为。如果中断持续时间超过了配置的容忍期（默认为五分钟），则断开连接的节点上的 Pod 将被标记为 `Terminating` 而新 Pod 则会被调度到可用区中运行良好的节点上。

## 实施 Amazon EKS 区域转移以提高弹性

与[Amazon 应用程序恢复控制器 \(ARC\) 集成的 Amazon EKS 区域转移](#)提供了一种在可用区受损期间主动管理流量的机制。此功能允许将网络流量从不健康的可用区临时重定向到该可用区内的健康区域，从而最大限度地减少服务中断。

## 了解区域偏移机制

Amazon EKS 区域转移可解决东西向流量（集群内的 Pod 间通信）。当使用应用程序负载均衡器或网络负载均衡器配置区域转移时，它还支持入口流量路由。该机制通过协调多个 Kubernetes 和 AWS 控制平面组件来运行，以在不中断正在运行的工作负载的情况下安全地重定向流量。在主动的区域转移期间，Amazon EKS 会自动执行以下协调操作：

- **节点封锁**：受损可用区中的所有节点都被封锁。这可以防止 Kubernetes 调度器在维护现有工作负载的同时在节点上放置新的 pod。

- **可用区重新平衡暂停**：对于托管节点组，可用区重新平衡操作将暂停，Auto Scaling 组将更新，以仅在运行良好的可用区中启动新的数据平面节点。这样可以确保受损区域中不会配置新的容量。
- **端点移除**：EndpointSlice 控制器从所有相关可用区中移除受损可用区中的 pod 端点 EndpointSlices。这样可以确保服务发现和负载平衡机制仅将流量路由到在运行良好的可用区中的 pod。
- **工作负载保留**：Amazon EKS 不会在受影响的可用区中终止节点或驱逐容器。它可以保持受损区域的全部容量，因此当区域转移到期或取消时，流量可以安全地返回，而无需额外的扩展操作。

## 区域移位激活方法

根据您的运营模式，您可以从两种方法中进行选择来启动区域转移：

- 当通过监控、警报或客户报告检测到特定的可用区问题时，[手动区域切换](#)可提供操作员驱动的控制。此方法需要通过 ARC 控制台 AWS Command Line Interface (AWS CLI) 或区域移位进行明确的操作 APIs，操作员可以在其中指定受损的可用区并定义班次的到期时间。当团队拥有专门的监控和待命能力，并且更愿意直接控制交通管理决策时，手动轮班是合适的。
- 当ARC根据内部遥测和多个可用区中的运行状况信号（包括网络指标、亚马逊弹性计算云（Amazon）和弹性负载平衡（Elastic Load Balancing EC2）AWS 服务，检测到潜在的可用区故障或损伤时，Zonal autoshift 授权 AWS 自动启动轮班。AWS 当指示器显示问题已解决时，自动结束自动换档。如果您希望以最少的人工干预实现最高的可用性，我们建议您使用这种方法，因为它可以对检测到的可用区损坏进行亚分钟响应。

## 有效区域偏移的先决条件

为了在可用区受损期间成功保护应用程序，在启用区域转移功能之前，您必须架构集群以实现多可用区弹性：

- **多可用区节点分布**：在至少三个可用区中配置工作节点，以确保在一个可用区不可用时有足够的冗余。
- **容量规划**：在运行正常的可用区中预配置足够的计算容量，以容纳一个可用区停止服务时的全部工作负载，因为在主动中断期间扩展操作可能会遇到容量不足的情况。
- **Pod 分发和预扩展**：[在所有可用区部署每个应用程序的多个副本，并在每个区域中预先扩展关键系统组件，例如 CoreDNS。](#) 这有助于确保在转移区域后仍有足够的容量。

## 关于区域干扰弹性的建议

- 在@@ 创建集群时启用区域切换：对于新的 EKS 集群，在初始配置期间通过 Amazon EKS 控制台或基础设施即代码 (IaC) 工具（例如）启用与 ARC 的区域转移集成。AWS CLI AWS CloudFormation 默认情况下，使用快速配置创建的 [EKS 自动模式集群](#) 会启用区域偏移。
- 选择适当的激活方法：对于需要通过自动响应实现最大可用性的生产环境，请选择区域自动切换，特别是对于面向客户的应用程序，在可用区受损期间停机几分钟可能会对业务产生重大影响。对于运营团队倾向于在交通转移之前提供明确批准的环境，或者应用程序测试和验证仍在进行中的环境，请使用手动分区切换。
- 在生产部署之前测试弹性：通过手动启动测试区域转移或启用区域自动移位实践运行来验证集群在单可用区丢失下的行为，以验证应用程序是否保持可用性、性能是否可以接受，并且在可用区数量减少的情况下运行时容量是否充足。我们强烈建议您进行此项测试，这样您就可以在可用区出现实际损坏之前发现配置差距。
- 协调负载均衡器配置：对于接收外部流量的应用程序，请在关联的应用程序负载均衡器和网络负载均衡器上启用 ARC 区域切换，以确保在可用区域受损期间，入口流量和集群内东西向流量同时移动。这种协调可以防止出现外部请求到达健康的 pod 但这些 pod 无法与移开区域中的依赖项通信的情况。
- 监控轮班操作：启用区域班次后，配置轮班事件的监控和警报，包括自动换班激活、手动轮班启动和轮班到期，以保持对交通管理操作及其对应用程序行为的影响的运营可见性。

## 轮班完成和恢复

当区域转移根据其配置的持续时间过期，或者在可用区缺陷解决后被手动取消时，EndpointSlice 控制器会自动更新所有区域 EndpointSlices 以重新合并已恢复的可用区域中的端点。当客户端刷新端点信息并建立新的连接时，流量会逐渐返回到先前受影响的区域。这无需手动干预或重新调度 pod 即可实现集群容量的充分利用。

# 结论

在设计架构以实现高应用程序可用性和弹性时，请考虑以下组件：

- 微服务应用程序（其 pod 和容器）
- 工作负载数据平面（入口控制器、容器、[Amazon VPC CNI](#) 等系统组件、服务网格 sidecar 和 kube-proxy）
- 工作负载管理层（控制器、准入控制器、网络策略引擎和这些组件的永久数据存储）
- Kubernetes 控制平面
- 基础架构（节点、网络和网络设备）

要解决这些组件注意事项，请使用以下关键策略：

- 为帮助确保高可用性和容错能力，请将工作负载分散到节点和可用区域中。
- 要保护关键工作负载，请使用 pod 中断预算（PDBs）在中断期间保持应用程序的稳定性。
- 为了帮助确保 Pod 正常运行并提供流量，请配置启动探针、存活探针、就绪探测器和负载均衡器运行状况检查。
- 要有效地管理容器状态转换，请配置容器生命周期挂钩。
- 要在节点故障或维护期间控制驱逐过程，请配置 Pod 驱逐时间。

通过实施这些实践，您可以显著增强在 Amazon EKS 上运行的应用程序的可靠性和弹性，从而确保强大的性能和高可用性。

# Resources ( 资源 )

- [Kubernetes Pod 拓扑分布限制](#) ( Kubernetes 文档 )
- [Karpenter FAQs](#) ( Karpenter 文档 )
- [Kubernet@@ es 的解调器 \( 存储库 \)](#) GitHub
- [可用性和 Pod 生命周期](#) Amazon EKS 最佳实践指南
- [优雅地关闭应用程序](#) Amazon EKS 最佳实践指南
- [\[EKS\] \[请求\] : 配置能力 pod-eviction-timeout和解决方法](#) ( 容器路线图存储库 )

## 文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 [RSS 源](#)。

变更	说明	日期
<a href="#">更新</a>	修订了关于在 <u>区域中断期间驱逐吊舱</u> 的部分。	2025 年 10 月 29 日
<a href="#">更新</a>	修订了“ <a href="#">使用 pod 拓扑分布约束</a> ”部分。	2025 年 1 月 27 日
<a href="#">初次发布</a>	—	2024 年 10 月 23 日

# AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

## 数字

### 7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- 重构/重新架构 - 充分利用云原生功能来提高敏捷性、性能和可扩展性，以迁移应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将您的本地 Oracle 数据库迁移到兼容 Amazon Aurora PostgreSQL 的版本。
- 更换平台 - 将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：在中将您的本地 Oracle 数据库迁移到适用于 Oracle 的亚马逊关系数据库服务 (Amazon RDS) AWS 云。
- 重新购买 - 转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将您的客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- 更换主机（直接迁移）- 将应用程序迁移到云中，无需进行任何更改即可利用云功能。示例：在中的 EC2 实例上将您的本地 Oracle 数据库迁移到 Oracle AWS 云。
- 重新定位（虚拟机监控器级直接迁移）：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您可以将服务器从本地平台迁移到同一平台的云服务。示例：将 Microsoft Hyper-V 应用程序迁移到 AWS。
- 保留（重访）- 将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用 - 停用或删除源环境中不再需要的应用程序。

## A

### ABAC

请参阅[基于属性的访问控制](#)。

### 抽象服务

参见[托管服务](#)。

## ACID

参见[原子性、一致性、隔离性、耐久性](#)。

### 主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。与[主动-被动迁移](#)相比，它更灵活，但需要更多的工作。

### 主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

## 聚合函数

一个 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括SUM和MAX。

## AI

参见[人工智能](#)。

## AIOps

参见[人工智能操作](#)。

## 匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

## 反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

## 应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

## 应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是[产品组合发现和分析过程](#)的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

## 人工智能 (AI)

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅[什么是人工智能？](#)

## 人工智能操作 (AIOps)

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AIOps AWS 迁移策略中使用的更多信息，请参阅[操作集成指南](#)。

## 非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

## 原子性、一致性、隔离性、持久性 (ACID)

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

## 基于属性的访问权限控制 (ABAC)

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (I [AM](#)) 文档 [AWS 中的 AB AC](#)。

## 权威数据源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

## 可用区

中的一个不同位置 AWS 区域，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

## AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF 将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人员角度针对的是负责人力资源 (HR)、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅[AWS CAF 网站](#)和[AWS CAF 白皮书](#)。

## AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。 AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

## B

### 坏机器人

旨在破坏个人或组织或对其造成伤害的机器人。

### BCP

参见[业务连续性计划](#)。

### 行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的[行为图中的数据](#)。

### 大端序系统

一个先存储最高有效字节的系统。另请参见[字节顺序](#)。

### 二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

### bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

### 蓝/绿部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前的应用程序版本（蓝色），在另一个环境中运行新的应用程序版本（绿色）。此策略可帮助您在影响最小的情况下快速回滚。

### 自动程序

一种通过互联网运行自动任务并模拟人类活动或互动的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的网络爬虫。其他一些被称为恶意机器人的机器人旨在破坏个人或组织或对其造成伤害。

## 僵尸网络

被恶意软件感染并受单方（称为机器人牧民或机器人操作员）控制的机器人网络。僵尸网络是最著名的扩展机器人及其影响力的机制。

## 分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅[关于分支](#)（GitHub 文档）。

## 破碎的玻璃通道

在特殊情况下，通过批准的流程，用户 AWS 账户可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅[Well-Architected 指南中的“实施破碎玻璃程序”](#)指示 AWS 器。

## 棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和[全新](#)策略混合。

## 缓冲区缓存

存储最常访问的数据的内存区域。

## 业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅[在 AWS 上运行容器化微服务](#)白皮书中的[围绕业务能力进行组织](#)部分。

## 业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

# C

## CAF

参见[AWS 云采用框架](#)。

## 金丝雀部署

向最终用户缓慢而渐进地发布版本。当您确信时，可以部署新版本并全部替换当前版本。

## CCoE

参见[云卓越中心](#)。

## CDC

请参阅[变更数据捕获](#)。

### 更改数据捕获 (CDC)

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

## 混沌工程

故意引入故障或破坏性事件来测试系统的弹性。您可以使用[AWS Fault Injection Service \(AWS FIS\)](#) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

## CI/CD

查看[持续集成和持续交付](#)。

## 分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

## 客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

## 云卓越中心 (CCoE)

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS 云企业战略博客上的[CCoE 帖子](#)。

## 云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常与[边缘计算](#)技术相关。

## 云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅[构建您的云运营模型](#)。

## 云采用阶段

组织迁移到以下阶段时通常会经历四个阶段 AWS 云：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 — 进行基础投资以扩大云采用率（例如，创建着陆区、定义 CCoE、建立运营模型）

- **迁移** - 迁移单个应用程序
- **重塑** - 优化产品和服务，在云中创新

Stephen Orban 在 AWS 云企业战略博客的博客文章 [《云优先之旅和采用阶段》](#) 中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅 [迁移准备指南](#)。

## CMDB

参见 [配置管理数据库](#)。

## 代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 Bitbucket Cloud。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD 管道可以使用多个存储库。

## 冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

## 冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

## 计算机视觉 (CV)

[人工智能](#) 领域，使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如，Amazon SageMaker AI 为 CV 提供了图像处理算法。

## 配置偏差

对于工作负载，配置会从预期状态发生变化。这可能会导致工作负载变得不合规，而且通常是渐进的，不是故意的。

## 配置管理数据库 ( CMDB )

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

## 合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的 [一致性包](#)。

## 持续集成和持续交付 ( CI/CD )

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。CI/CD 通常被描述为管道。CI/CD 可以帮助您实现流程自动化、提高生产力、提高代码质量和更快地交付。有关更多信息，请参阅[持续交付的优势](#)。CD 也可以表示持续部署。有关更多信息，请参阅[持续交付与持续部署](#)。

## CV

参见[计算机视觉](#)。

## D

### 静态数据

网络中静止的数据，例如存储中的数据。

### 数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是 Well-Architected Framework 中安全支柱的一个组成部分。有关详细信息，请参阅[数据分类](#)。

### 数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

### 传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

### 数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

### 数据最小化

仅收集并处理绝对必要数据的原则。在 AWS 云中进行数据最小化可以降低隐私风险、成本和分析碳足迹。

### 数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅[上构建数据边界](#)。AWS

## 数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

## 数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

## 数据主体

正在收集和处理其数据的个人。

## 数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

## 数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

## 数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

## DDL

参见[数据库定义语言](#)。

## 深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

## 深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

## defense-in-depth

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，一种 defense-in-depth 方法可以结合多因素身份验证、网络分段和加密。

## 委托管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中 [使用 AWS Organizations 的服务](#)。

## 后

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

## 开发环境

参见 [环境](#)。

## 侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出警报。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS 上实施安全控制中的 [侦测性控制](#)。

## 开发价值流映射 (DVSM)

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

## 数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

## 维度表

在 [星型架构](#) 中，一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

## 灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

## 灾难恢复 (DR)

您用来最大限度地减少 [灾难](#) 造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅 Well-Architected Framework AWS work 中的 “[工作负载灾难恢复：云端 AWS 恢复](#)”。

## DML

参见[数据库操作语言](#)。

## 领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。Eric Evans 在其著作领域驱动设计：软件核心复杂性应对之道 ( Boston: Addison-Wesley Professional, 2003 ) 中介绍了这一概念。有关如何将领域驱动设计与 strangler fig 模式结合使用的信息，请参阅[使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET \( ASMX \) Web 服务现代化](#)。

## DR

参见[灾难恢复](#)。

## 漂移检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来[检测系统资源中的偏差](#)，也可以使用 AWS Control Tower 来[检测着陆区中可能影响监管要求合规性的变化](#)。

## DVSM

参见[开发价值流映射](#)。

## E

## EDA

参见[探索性数据分析](#)。

## EDI

参见[电子数据交换](#)。

## 边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与[云计算](#)相比，边缘计算可以减少通信延迟并缩短响应时间。

## 电子数据交换 (EDI)

组织之间自动交换业务文档。有关更多信息，请参阅[什么是电子数据交换](#)。

## 加密

一种将人类可读的纯文本数据转换为密文的计算过程。

### 加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

### 字节顺序

字节在计算机内存中的存储顺序。大端序系统先存储最高有效字节。小端序系统先存储最低有效字节。

## 端点

参见 [服务端点](#)。

### 端点服务

一种可以在虚拟私有云 (VPC) 中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务，AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud (Amazon VPC) 文档中的[创建端点服务](#)。

### 企业资源规划 (ERP)

一种自动化和管理企业关键业务流程（例如会计、[MES](#) 和项目管理）的系统。

### 信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的[信封加密](#)。

## 环境

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。
- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

## epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如，AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅[计划实施指南](#)。

## ERP

参见[企业资源规划](#)。

## 探索性数据分析 (EDA)

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

# F

## 事实表

[星形架构](#)中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

## 失败得很快

一种使用频繁和增量测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

## 故障隔离边界

在中 AWS 云，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅[AWS 故障隔离边界](#)。

## 功能分支

参见[分支](#)。

## 特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

## 特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释 (SHAP) 和积分梯度。有关更多信息，请参阅使用[机器学习模型的可解释性 AWS](#)。

## 功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

## 少量提示

在要求法学硕士执行类似任务之前，向其提供少量示例，以演示该任务和所需的输出。这种技术是情境学习的应用，模型可以从提示中嵌入的示例（镜头）中学习。对于需要特定格式、推理或领域知识的任务，Few-shot 提示可能非常有效。另请参见零镜头提示。

## FGAC

请参阅[精细的访问控制](#)。

### 精细访问控制 (FGAC)

使用多个条件允许或拒绝访问请求。

## 快闪迁移

一种数据库迁移方法，它使用连续的数据复制，通过[更改数据捕获](#)在尽可能短的时间内迁移数据，而不是使用分阶段的方法。目标是将停机时间降至最低。

## FM

参见[基础模型](#)。

### 基础模型 (FM)

一个大型深度学习神经网络，一直在广义和未标记数据的大量数据集上进行训练。FMs 能够执行各种各样的一般任务，例如理解语言、生成文本和图像以及用自然语言进行对话。有关更多信息，请参阅[什么是基础模型](#)。

## G

### 生成式人工智能

[人工智能](#)模型的子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和工件，例如图像、视频、文本和音频。有关更多信息，请参阅[什么是生成式 AI](#)。

## 地理封锁

请参阅[地理限制](#)。

## 地理限制 ( 地理阻止 )

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档[中的限制内容的地理分布](#)。

## GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的，而[基于主干的工作流程](#)是现代的首选方法。

## 金色影像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上配置软件，并有助于提高设备制造运营的速度、可扩展性和生产力。

## 全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为[棕地](#)）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

## 防护机制

帮助管理各组织单位的资源、策略和合规性的高级规则 (OUs)。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性防护机制会检测策略违规和合规性问题，并生成警报以进行修复。它们通过使用 AWS Config、Amazon、AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

## H

### HA

参见[高可用性](#)。

## 异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。[AWS 提供了 AWS SCT](#) 来帮助实现架构转换。

## 高可用性 (HA)

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

## 历史数据库现代化

一种用于实现运营技术 (OT) 系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

## 抵制数据

从用于训练[机器学习](#)模型的数据集中扣留的一部分带有标签的历史数据。通过将模型预测与抵制数据进行比较，您可以使用抵制数据来评估模型性能。

## 同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

## 热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

## 修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

## hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

## 我

## IaC

参见[基础设施即代码](#)。

## 基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS 云环境中的权限。

## 空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将它们保留在本地。

## IIoT

参见 [工业物联网](#)。

## 不可变的基础架构

一种为生产工作负载部署新基础架构，而不是更新、修补或修改现有基础架构的模型。[不可变基础设施本质上比可变基础架构更一致、更可靠、更可预测](#)。有关更多信息，请参阅 Well-Architected Framework 中的 [使用不可变基础架构 AWS 部署最佳实践](#)。

## 入站（入口）VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## 增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

## 工业 4.0

该术语由[克劳斯·施瓦布 \(Klaus Schwab\)](#)于2016年推出，指的是通过连接、实时数据、自动化、分析和人工智能/机器学习的进步实现制造流程的现代化。

## 基础设施

应用程序环境中包含的所有资源和资产。

## 基础设施即代码 (IaC)

通过一组配置文件预置和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

## 工业物联网 (IIoT)

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅[制定工业物联网 \(IIoT\) 数字化转型战略](#)。

## 检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理对 VPCs（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量的检查。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## 物联网 (IoT)

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅[什么是 IoT？](#)

## 可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅使用[机器学习模型的可解释性 AWS](#)。

## IoT

参见[物联网](#)。

## IT 信息库 (ITIL)

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

## IT 服务管理 (ITSM)

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅[运营集成指南](#)。

## ITIL

请参阅[IT 信息库](#)。

## ITSM

请参阅[IT 服务管理](#)。

## L

## 基于标签的访问控制 (LBAC)

强制访问控制 (MAC) 的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

## 登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅[设置安全且可扩展的多账户 AWS 环境](#)。

## 大型语言模型 (LLM)

一种基于大量数据进行预训练的深度学习 [AI](#) 模型。法学硕士可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅[什么是 LLMs](#)。

## 大规模迁移

迁移 300 台或更多服务器。

## LBAC

请参阅[基于标签的访问控制](#)。

## 最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的[应用最低权限许可](#)。

## 直接迁移

见 [7 R](#)。

## 小端序系统

一个先存储最低有效字节的系统。另请参见[字节顺序](#)。

## LLM

参见[大型语言模型](#)。

## 下层环境

参见[环境](#)。

## M

## 机器学习 ( ML )

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网 ( IoT ) 数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅[机器学习](#)。

## 主分支

参见[分支](#)。

## 恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问。恶意软件的示例包括病毒、蠕虫、勒索软件、特洛伊木马、间谍软件和键盘记录器。

## 托管服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。亚马逊简单存储服务 (Amazon S3) Service 和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

## 制造执行系统 (MES)

一种软件系统，用于跟踪、监控、记录和控制在车间将原材料转化为成品的生产过程。

## MAP

参见[迁移加速计划](#)。

## 机制

一个完整的过程，在此过程中，您可以创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运行过程中自我增强和改进的循环。有关更多信息，请参阅在 Well-Architect AWS ed 框架中[构建机制](#)。

## 成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

## MES

参见[制造执行系统](#)。

## 消息队列遥测传输 (MQTT)

一种基于发布/订阅模式的轻量级 machine-to-machine (M2M) 通信协议，适用于资源受限的物联网设备。

## 微服务

一种小型的独立服务，通过明确的定义进行通信 APIs，通常由小型的独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务

的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅[使用 AWS 无服务器服务集成微服务。](#)

## 微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级通过定义明确的接口进行通信。 APIs 该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅[在上实现微服务。 AWS](#)

## 迁移加速计划（MAP）

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。 MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

## 大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 [AWS 迁移策略](#) 的第三阶段。

## 迁移工厂

跨职能团队，通过自动化、敏捷的方法简化工作负载迁移。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发 DevOps 人员和冲刺专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中[有关迁移工厂的讨论](#) 和 [云迁移工厂](#) 指南。

## 迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

## 迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：EC2 使用 AWS 应用程序迁移服务重新托管向 Amazon 的迁移。

## 迁移组合评测（MPA）

一种在线工具，可提供信息，用于验证迁移到的业务案例。 AWS 云 MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用 [MPA 工具](#)（需要登录）。

## 迁移准备情况评测 ( MRA )

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅[迁移准备指南](#)。MRA 是 [AWS 迁移策略](#)的第一阶段。

## 迁移策略

用于将工作负载迁移到的方法 AWS 云。有关更多信息，请参阅此词汇表中的[7R](#) 条目和[动员组织以加快大规模迁移](#)。

## ML

参见[机器学习](#)。

## 现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅[中的应用程序现代化策略](#)。[AWS 云](#)

## 现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅[中的评估应用程序的现代化准备情况](#)。[AWS 云](#)。

## 单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅[将单体分解为微服务](#)。

## MPA

参见[迁移组合评估](#)。

## MQTT

请参阅[消息队列遥测传输](#)。

## 多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

## 可变基础架构

一种用于更新和修改现有生产工作负载基础架构的模型。为了提高一致性、可靠性和可预测性，Well-Architect AWS ed Framework 建议使用[不可变基础设施](#)作为最佳实践。

## O

### OAC

请参阅[源站访问控制](#)。

### OAI

参见[源访问身份](#)。

### OCM

参见[组织变更管理](#)。

## 离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

## OI

参见[运营集成](#)。

### OLA

参见[运营层协议](#)。

## 在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

### OPC-UA

参见[开放流程通信-统一架构](#)。

### 开放流程通信-统一架构 (OPC-UA)

一种用于工业自动化的 machine-to-machine ( M2M ) 通信协议。OPC-UA 提供了数据加密、身份验证和授权方案的互操作性标准。

## 运营级别协议 (OLA)

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议 (SLA)。

## 运营准备情况审查 (ORR)

一份问题清单和相关的最佳实践，可帮助您理解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 Well-Architect AWS Framework 中的 [运营准备情况评估 \(ORR\)](#)。

## 操作技术 (OT)

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术 (IT) 系统的集成是 [工业 4.0](#) 转型的重点。

## 运营整合 (OI)

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅 [运营整合指南](#)。

## 组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail 文档中的 [为组织创建跟踪](#)。

## 组织变革管理 (OCM)

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 [OCM 指南](#)。

## 来源访问控制 (OAC)

中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

## 来源访问身份 (OAI)

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 [OAC](#)，其中提供了更精细和增强的访问控制。

## ORR

参见[运营准备情况审查](#)。

## OT

参见[运营技术](#)。

## 出站（出口）VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。[AWS 安全参考架构](#)建议设置您的网络帐户，包括入站、出站和检查，VPCs 以保护您的应用程序与更广泛的互联网之间的双向接口。

## P

### 权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的[权限边界](#)。

### 个人身份信息 (PII)

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

### PII

查看[个人身份信息](#)。

### playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

### PLC

参见[可编程逻辑控制器](#)。

### PLM

参见[产品生命周期管理](#)。

### policy

一个对象，可以在中定义权限（参见[基于身份的策略](#)）、指定访问条件（参见[基于资源的策略](#)）或定义组织中所有账户的最大权限 AWS Organizations（参见[服务控制策略](#)）。

## 多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。有关更多信息，请参阅[在微服务中实现数据持久性](#)。

## 组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅[评估迁移准备情况](#)。

## 谓词

返回true或的查询条件false，通常位于子WHERE句中。

## 谓词下推

一种数据库查询优化技术，可在传输前筛选查询中的数据。这减少了必须从关系数据库检索和处理的数据量，并提高了查询性能。

## 预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS 上实施安全控制中的[预防性控制](#)。

## 主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。AWS 账户有关更多信息，请参阅 IAM 文档中[角色术语和概念](#)中的主体。

## 通过设计保护隐私

一种在整个开发过程中考虑隐私的系统工程方法。

## 私有托管区

一个容器，其中包含有关您希望 Amazon Route 53 如何响应针对一个或多个 VPCs 域名及其子域名的 DNS 查询的信息。有关更多信息，请参阅 Route 53 文档中的[私有托管区的使用](#)。

## 主动控制

一种[安全控制措施](#)，旨在防止部署不合规的资源。这些控件会在资源配置之前对其进行扫描。如果资源与控件不兼容，则不会对其进行配置。有关更多信息，请参阅 AWS Control Tower 文档中的[控制参考指南](#)，并参见在上实施安全[控制](#)中的主动控制 AWS。

## 产品生命周期管理 (PLM)

在产品的整个生命周期中，从设计、开发和上市，到成长和成熟，再到衰落和移除，对产品进行数据和流程的管理。

### 生产环境

参见[环境](#)。

### 可编程逻辑控制器 (PLC)

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

### 提示链接

使用一个[LLM](#) 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

### 假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

### publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，以提高可扩展性和响应能力。例如，在基于微服务的[MES](#)中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

## Q

### 查询计划

一系列步骤，例如指令，用于访问 SQL 关系数据库系统中的数据。

### 查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

# R

## RACI 矩阵

参见“[负责任、负责、咨询、知情”\(RACI\)](#)。

## RAG

请参见[检索增强生成](#)。

## 勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

## RASCI 矩阵

参见“[负责任、负责、咨询、知情”\(RACI\)](#)。

## RCAC

请参阅[行和列访问控制](#)。

## 只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

## 重新架构师

见[7 R](#)。

## 恢复点目标 (RPO)

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

## 恢复时间目标 (RTO)

服务中断和服务恢复之间可接受的最大延迟。

## 重构

见[7 R](#)。

## Region

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，彼此独立，以提供容错、稳定性和弹性。有关更多信息，请参阅[指定 AWS 区域 您的账户可以使用的账户](#)。

## 回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

## 重新托管

见 [7 R](#)。

## 版本

在部署过程中，推动生产环境变更的行为。

## 搬迁

见 [7 R](#)。

## 更换平台

见 [7 R](#)。

## 回购

见 [7 R](#)。

## 故障恢复能力

应用程序抵御中断或从中断中恢复的能力。在中规划弹性时，[高可用性](#)和[灾难恢复](#)是常见的考虑因素。AWS 云有关更多信息，请参阅[AWS 云弹性](#)。

## 基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

## 责任、问责、咨询和知情 (RACI) 矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责 (R)、问责 (A)、咨询 (C) 和知情 (I)。支持 (S) 类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

## 响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在 AWS 上实施安全控制中的[响应性控制](#)。

## 保留

见 [7 R](#)。

## 退休

见 [7 R](#)。

## 检索增强生成 ( RAG )

一种[生成式人工智能](#)技术，其中[法学硕士](#)在生成响应之前引用其训练数据源之外的权威数据源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅[什么是 RAG](#)。

## 轮换

定期更新[密钥](#)以使攻击者更难访问凭据的过程。

## 行列访问控制 (RCAC)

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

## RPO

参见[恢复点目标](#)。

## RTO

参见[恢复时间目标](#)。

## 运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

## S

## SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS 管理控制台 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的[关于基于 SAML 2.0 的联合身份验证](#)。

## SCADA

参见[监督控制和数据采集](#)。

## SCP

参见[服务控制政策](#)。

## secret

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 [Secrets Manager 密钥中有什么？](#) 在 Secrets Manager 文档中。

## 安全性源于设计

一种在整个开发过程中考虑安全性的系统工程方法。

## 安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制主要有四种类型：[预防性](#)、[侦测](#)、[响应式](#)和[主动](#)式。

## 安全加固

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

## 安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

## 安全响应自动化

一种预定义和编程的操作，旨在自动响应或修复安全事件。这些自动化可作为[侦探或响应式](#)安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换证书。

## 服务器端加密

在目的地对数据进行加密，由接收方 AWS 服务 进行加密。

## 服务控制策略（SCP）

一种策略，用于集中控制组织中所有账户的权限 AWS Organizations。SCPs 定义防护措施或限制管理员可以委托给用户或角色的操作。您可以使用 SCPs 允许列表或拒绝列表来指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的[服务控制策略](#)。

## 服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 [AWS 服务 端点](#)。

## 服务水平协议 (SLA)

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

### 服务级别指示器 (SLI)

对服务性能方面的衡量，例如其错误率、可用性或吞吐量。

### 服务级别目标 (SLO)

代表服务运行状况的目标指标，由服务[级别指标](#)衡量。

### 责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。 AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅[责任共担模式](#)。

### SIEM

参见[安全信息和事件管理系统](#)。

### 单点故障 (SPOF)

应用程序的单个关键组件出现故障，可能会中断系统。

### SLA

参见[服务级别协议](#)。

### SLI

参见[服务级别指标](#)。

### SLO

参见[服务级别目标](#)。

### split-and-seed 模型

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅[中的分阶段实现应用程序现代化的方法。 AWS 云](#)

### 恶作剧

参见[单点故障](#)。

### 星型架构

一种数据库组织结构，它使用一个大型事实表来存储交易数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在[数据仓库](#)中使用或用于商业智能目的而设计。

## strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 [Martin Fowler](#) 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅[使用容器和 Amazon API Gateway 逐步将原有的 Microsoft ASP.NET \( ASMX \) Web 服务现代化。](#)

## 子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

## 监控和数据采集 (SCADA)

在制造业中，一种使用硬件和软件来监控有形资产和生产操作的系统。

## 对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

## 综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 [Amazon S CloudWatch Synthetics](#) 来创建这些测试。

## 系统提示符

一种向法学硕士提供上下文、说明或指导方针以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

# T

## 标签

键值对，充当用于组织资源的元数据。 AWS 标签有助于您管理、识别、组织、搜索和筛选 资源。有关更多信息，请参阅[标记您的 AWS 资源。](#)

## 目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

## 任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

## 测试环境

参见[环境](#)。

## 训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

## 中转网关

一个网络传输中心，可用于将您的网络 VPCs 和本地网络互连。有关更多信息，请参阅 AWS Transit Gateway 文档中的[什么是公交网关](#)。

## 基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

## 可信访问权限

向您指定的服务授予权限，该服务可代表您在其账户中执行任务。AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的[AWS Organizations 与其他 AWS 服务一起使用](#)。

## 优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

## 双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

## U

## 不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。有关更多信息，请参阅[量化深度学习系统中的不确定性](#)指南。

## 无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

## 上层环境

参见[环境](#)。

## V

### vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

## 版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

## VPC 对等连接

两者之间的连接 VPCs，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的[什么是 VPC 对等连接](#)。

## 漏洞

损害系统安全的软件缺陷或硬件缺陷。

## W

### 热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

### 暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

### 窗口函数

一个 SQL 函数，用于对一组以某种方式与当前记录相关的行进行计算。窗口函数对于处理任务很有用，例如计算移动平均线或根据当前行的相对位置访问行的值。

## 工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

## 工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

## 蠕虫

参见[一次写入，多读](#)。

## WQF

请参阅[AWS 工作负载资格框架](#)。

## 一次写入，多次读取 (WORM)

一种存储模型，它可以一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但他们无法对其进行更改。这种数据存储基础架构被认为是[不可变](#)的。

## Z

### 零日漏洞利用

一种利用未修补[漏洞](#)的攻击，通常是恶意软件。

### 零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

### 零镜头提示

向[法学硕士](#)提供执行任务的说明，但没有示例（镜头）可以帮助指导任务。法学硕士必须使用其预先训练的知识来处理任务。零镜头提示的有效性取决于任务的复杂性和提示的质量。另请参阅[few-shot 提示](#)。

### 僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。