**挑战和问题：**

在像Microsoft Azure这样的云规模系统基础设施中，不同的团队为了部署新特性、修复现有bug、调优性能等 需要频繁地在代码和配置上进行更改。

但是此类基础设施的规模大和复杂性高，即使在更新组件时出现一个小问题，也会导致广泛的故障，对客户造成严重的影响。

虽然实际生产中会经过严格的评审和测试，但是由于测试和生产环境在集群大小、OS版本、工作负载等方面的差异，一些bug可能没有被发现。所以准确评估云系统中部署的影响具有挑战性。

假阴性：漏掉一些潜在的问题

假警报：所有这些事件都很容易误导本地部署运行状况监视器，使其错误地将故障归因于无害的部署。这些错误的警报将导致无害的发布停止，并阻止及时应用更改。而且开发人员在调查此类错误警报方面也浪费了大量的时间和资源。

**解决方案：**

Gandalf：一种用于安全部署云基础设施的端到端分析服务。

它解决了前面提到的挑战，用来确保云基础设施的安全部署。

Gandalf没有根据单个组件日志分别分析每个部署，而是采用自上而下的方法来全面评估部署的影响。

Gandalf连续监控来自基础设施遥测数据的丰富信号集，包括服务级别日志、性能计数器和进程级事件。

当检测到系统异常时，Gandalf会分析它是否是由部署引起的。如果发现一个不好的部署就会阻止它。

Gandalf设计了一个lambda体系结构的Gandalf系统[6]，将实时决策引擎与批决策引擎相结合。

Gandalf还提供了详细的支持证据和交互式前端，以便工程师轻松理解问题和根本原因。

Gandalf的核心决策逻辑是一个由异常检测、相关分析和影响评估组成的新模型。

该模型首先从原始遥测数据中检测异常。

然后，它通过时间和空间相关性以及集合排序算法来确定卷展是否与检测到的故障高度相关。

最后，该模型使用高斯判别分类器来判断可疑卷展所造成的影响是否足以停止部署。

**系统设计：**

Data Sourc，流和批处理，结果编排和操作，监测和诊断前端

1. **Data Source:**

Gandalf处理三种类型的数据：

（1）performance data性能数据，如节点中的CPU性能计数器和内存使用情况；

（2）failure signals故障信号，如代理故障、容器故障、操作系统崩溃、节点重新启动和API调用异常；

（3）update events 描述组件部署信息的更新事件。

Gandalf使用来自各种数据源的综合信号

对于某些信号，Gandalf执行预处理以解析原始数据（例如，日志消息）并提取失败签名（例如，错误代码）。

在分析过程中，这些预处理的信号根据它们的时间戳、节点id和服务类型进行聚合。

1. **Stream and Batch Processing:**

为了平衡速度和覆盖率，Gandalf采用lambda体系结构，同时具有流式和批处理分析引擎

speed layer使用来自快速管道Microsoft Kusto[7]的数据,Kusto有一种基于数据流模型的定制查询语言。Kusto有很短的延迟，但是它不能有效地处理大量的数据。

批处理层使用来自Cosmos的数据,Cosmos是一个类似Hadoop的文件系统，数据源延迟长达数小时。尽管延迟相对较长，COSMOS能处理非常大的数据量

lambda架构允许我们提供快速决策和更高的覆盖率。

1. **Result Orchestration and Actions :**

Gandalf的服务层使用Azure服务结构框架[8]实现为一个高可靠性和可伸缩的web服务。

在每次运行Gandalf的异常检测和相关算法（将在第4节中描述）之后，来自速度层和批处理层的结果通过web服务存储在两个单独的报告表中。这些结果包括部署影响评估、建议的决策、异常模式、相关信息等。

各种DevOps应用程序从报告表中提取结果。

在应用程序中，最重要的是通知服务。当通知服务通知到一个新的禁止决定时，它会向拥有该决定的团队发送一封关于该决定的电子邮件，并创建包含详细信息的事件通知单。它还通过键值存储通知部署引擎批准或停止部署。

1. **Monitoring and Diagnosis Front-End :**

提供了一个web前端，为开发人员提供了实时的发布监控和问题诊断支持。

**算法设计：**

相关性模型的步骤：

（1）异常检测从原始遥测数据中检测系统级故障；

（2）相关性分析确定在多个展开中对检测到的故障负责的组件；又分为四个部分，即集合投票、时空相关和指数时间衰减。

（3）决策步骤评估受影响的范围，并决定是否应停止部署。

1. 异常检测：

首先使用经验日志解析器，用虚拟标识符替换诸如vm id、subscription ID等唯一标识符。

然后，运行一个简化的增量式层次聚类模型[36]，将所有处理过的文本分组为一组错误模式。在获得故障特征后，Gandalf根据每个故障特征的出现情况来检测异常。

Gandalf根据过去的数据，利用Holt-Winters预测[14]来检测异常情况。

1. 相关性分析：

一旦检测到异常，Gandalf需要将观察到的故障与部署事件关联起来，并动态评估故障的影响。由于许多组件是同时部署的，所以使用投票否决机制来建立故障和部署组件之间的关系。

1.全体投票。Ensemble Voting

对于发生在时间戳tf上的故障e和在时间戳td上部署在同一结点上的rollout组件c，

每个故障e在一个窗口大小w\_b内为它之前部署的所有组件(即td< tf)投票

并且否决在它之后在窗口大小w\_a内部署的所有组件(即td> tf)。

2.时空相关性

计算时间相关性得分

空间相关性进行评价

3.时间衰减

后面出现的故障对前面故障的权重衰减，随着时间

1. 决策过程

通过评估部署的影响范围（如受影响的集群数量、受影响的节点数量、受影响的客户数量等）来为组件cj做出“通过/不通过”的决策，

决策准则用高斯判别分类器动态训练[9]。

训练数据是根据历史部署案例和组件团队的反馈生成的。

1. 结合领域知识

默认情况下，Gandalf平等对待输入的故障信号，但也允许开发人员使用可定制的权重来指定某些故障的重要性。权重是从0到100的相对值，表示最不重要到最重要。故障信号的默认权重为1。

**实验：**

Gandalf已经在Azure产品上运行了18个多月。

Gandalf系统平均每天处理270K个平台事件，高峰日处理77000个事件，每天在控制平面中记录约6亿个API调用，包括2000多种故障类型。每天分析的总数据量超过20TB。

对于每一个部署，对于每一个部署，Gandalf可以在5分钟左右的时间内做出决定，端到端的速度层，以及大约3个小时的批处理层。

，将整个生产机队的部署时间缩短了一半以上

Gandalf在进入生产前阻止了99.2%的不良推广。

对于数据平面的推出，Gandalf实现了92.4%的准确率和100%的召回率。

对于控制平面的展开，Gandalf实现了94.9%的精确度和99.8%的召回率。

**优缺点：**

优点：

丰富的数据集

流式处理和批处理相结合，达到检测速度和覆盖度上都达到比较好的效果

详细的报告结果和证据内容

支持前端，方便交互

缺点：可移植性是否良好未知。这篇文章主要使对 Azure 上部署，不知道在其他平台上部署

是否也能达到这么好的效果。