38 | GitOps 为什么成为云原生交付的事实标准？

2023-03-06 王炜 来自北京

《云原生架构与GitOps实战》 课程介绍

讲述：王炜

时长 08:58 大小 8.19M

你好，我是王炜。

这节课，我想和你分享一下 GitOps 的历史和发展过程。

时间回到 2017 年，一家做 Kubernetes 解决方案的初创公司 Weaveworks 首次提出了 GitOps，在那个 DevOps 盛行的年代，GitOps 绝对是具有创造性的。Weaveworks 对 GitOps 的定义是：利用云原生工具和云服务进行应用程序部署和管理的最佳实践，定位是 DevOps 的进一步扩展。

除了给出定义，Weaveworks 还开源了 FluxCD。没错，它就是现在和 ArgoCD 竞争的 CNCF 毕业项目。它们的作用都是监听 Git 仓库的变化，和集群内的对象进行对比，并自动应用有差异的部分。

不过，需要注意的是，GitOps 并不等于 FluxCD 或者 ArgoCD，它代表的是一种工程实践的方法。那么，为什么这个方法会成为应用交付的事实标准呢？到底要怎么理解 GitOps，相比较传统的交付过程，它独特的优势是什么？

首先，我们需要先理解 GitOps 到底是什么？

GitOps 是什么？

根据 Weaveworks 的总结，我们可以简单地把 GitOps 概括为下面两件事。

1. 它是一种管理模型，也是一种云原生技术，负责为应用程序的部署、管理和监控提供统一的最佳实践。
2. 它提供了一种实现开发者自助发布的路径，统一了开发团队和运维团队。

更进一步，GitOps 为开发者提供了持续部署的标准，它以开发者为中心，为开发者提供基础设施的管理和运维方法。它通过使用开发者已经熟悉的工具，例如 Git 来管理 Kubernetes 集群，实现应用交付。

Git 仓库承担了基础架构和应用程序“单一事实来源”的职责，通过 Git 仓库，开发者可以使用他们熟悉的推送、拉取和 Pull Request 流程来对基础架构和应用进行修改。

GitOps 的 4 个原则

不过，任何方案都需要有一个标准，为了建立行业内的 GitOps 标准，由 CNCF 牵头，2020 年 GitOps 工作组成立了，最初的工作组由 Weaveworks、微软、GitHub 和亚马逊等公司组成。作立工作组之后的第一步，他们启动了 OpenGitOps 项目，它目前是 CNCF Sandbox 项目。

此外，GitOps 工作组还制定了 GitOps 的基本原则，它们分别是：

1. 声明式
2. 版本化和不可变
3. 自动拉取
4. 持续协调

让我们来分别介绍一下这几个基本原则。

原则一：声明式

声明式是实现 GitOps 的核心基础。在 GitOps 中，所有工具必须是声明式的并将 Git 作为单一来源，应用程序可以非常方便地部署到 Kubernetes 集群中，最重要的是，当出现平台级故障时，你的应用可以随时部署到其他的标准平台上。

原则二：版本化和不可变

有了声明式的帮助，基础设施和应用的版本可以映射为 Git 源码对应的版本，你可以通过 Git Revert 随时进行回滚操作。更重要的是 Git 仓库不会随着时间的推移出现版本变化。

原则三：自动拉取

一旦将声明式的状态合并到 Git 仓库中，也就意味着提交会自动应用到集群中。这种部署方式安全且高效，部署过程无需人工干预，由于整个过程是自动化的，且不存在人工运行命令的步骤，这就杜绝了人为错误的可能性。当然，为了安全起见，你也可以在部署过程加入人工审批环节。

原则四：持续协调

“协调”过程实际上依赖于 FluxCD 或者 ArgoCD 这些控制器，它们会定期自动拉取 Git 仓库并对比它和集群的差异，然后将差异应用到集群中，这样可以确保整个系统能够进行自我修复。

GitOps 的优势

我们知道，当有新的内容提交到 Git 仓库的时候，GitOps 流水线会自动对基础设施和应用进行修改。但实际上，背后的机制比看起来要复杂得多。GitOps 工具能够自动对基础设施的状态以及 Git 代码仓库的定义进行对比，然后当状态不一致的时候提示并自动同步。

总结来说，GitOps 的优势主要体现在 5 个方面：

1. 提升发布效率
2. 优化开发者体验

3. 更高的稳定性和可靠性

4. 标准化和一致性

5. 更强的安全性

优势一：提升发布效率

GitOps 显著降低了软件发布所需要的时间。Weaveworks 估计，团队每天的发布次数提升了 30 到 100 倍，开发效率提升了 2-3 倍。

优势二：优化开发者体验

GitOps 流水线包含 CI 构建过程，对开发者屏蔽了 Kubernetes 内部复杂的工作原理，开发人员只需要熟悉 Git 的使用就可以间接控制 Kubernetes 的更新。此外，GitOps 也对新手工程师非常友好，降低了开发门槛。最后，GitOps 为开发者提供了自助式的发布体验，开发人员可以随时发布和回滚应用。

优势三：更高的稳定性和可靠性

因为 Git 是唯一可信源，它很容易进行回滚和恢复，所以当系统出现故障时，开发者只需要对 Git 仓库进行回滚即可，这将系统恢复时间从几小时缩短到了几分钟。此外，由于每次变更都会产生新的提交，相当于提供了一个审计日志，它详细记录了谁在什么时候对系统进行了什么修改，有助于日后追溯操作记录。

优势四：标准化和一致性

借助声明式和 Kubernetes，Git 仓库定义的对象都是标准化的，它天然支持不同的云厂商，当我们需要在其他云厂商重建环境时，只需要修改部署的目标集群即可。此外，GitOps 流水线对组织的所有团队来说都是一致的，这可以避免不同的小组在实现相同的部署需求时重复造轮子。

优势五：更强的安全性

由于 GitOps 借助 Git 来存储标准的定义文件，而 Git 的安全性又非常好，所以 GitOps 的存储过程也是安全的。此外，在开发者侧，并不会直接接触到基础设施的凭据，所以，相比较传统

的发布过程，GitOps 具有更高的安全性。

成为交付标准：GtiOp 的变革性影响

GitOps 之所以能成为云原生应用交付的标准，除了上述 5 大优势以外，还因为它给现有的 DevOps 应用交付模式带来了巨大变革。

这里所说的变革性影响主要包括下面几个方面。

1. 将应用交付从推模式转变为了拉模式。
2. 补充了 Infra As Code（基础设施即代码）的不足。
3. 逐渐取代了 DevOps 的位置。

接下来我们详细介绍一下这几个方面。

将应用交付从推模式转变为了拉模式

在 DevOps 主导的应用交付过程中，CD 工具往往需要在 CI 流水线执行完成之后才会启动。在这个过程中，CD 工具需要具有集群的访问权限。而在 GitOps 的流程中，当有新的变更提交到 Git 仓库时，集群内的 GitOps 工具会自动对比差异并执行变更。

那为什么应用交付从推模式变成了拉模式就产生了如此巨大的差异呢？

在推模式下，CI 或者 CD 修改集群对象时，它需要在集群外部取得集群的凭据，这是非常不安全的做法。此外，推模式下的部署过程往往是命令式的，例如通过 kubectl apply 来执行变更，由于缺少“协调”的过程，在变更过程中该行为并不是原子性的。

而 GitOps 通过 Operator 在集群内实现了拉的模式，在解决凭据安全问题的同时，也加入了“协调”过程，这个过程就像是一个不断运行的监视器，不断拉取仓库变更并对比差异，这一切都是在集群内实现的。

出于安全性和原子性考虑，应用交付从推模式正逐渐被拉模式取代。

补充了基础设施即代码的不足

以 Terraform 为代表的基础设施即代码获得了巨大的成功，它将以往需要通过命令式的操作变成了 HCL 声明式的配置方式。GitOps 继承了这个思想，在 GitOps 流程中，不仅能够通过声明式的方式定义基础设施，而且还可以定义应用的交付方式，弥补了基础设施即代码在定义交付方式时的不足。

逐渐取代了 DevOps 的位置

虽然 DevOps 比 GitOps 支持更广泛的应用程序模型，但随着容器化和 Kubernetes 技术的普及，DevOps 的优势已经不这么明显了。

其次，随着云原生技术的发展，DevOps 的工具链已经逐渐落后于 Kubernetes 的生态系统。GitOps 的工具链相对来说更加轻量，也更符合云原生快速发展的标准。

虽然 DevOps 和 GitOps 并不是完全独立的，它们有许多共同的目标。但随着云原生的普及，GitOps 和 Kubernetes 的组合注定会成为新的工程实践方式。而随着 GitOps 在开发者群体中的认可度越来越高，DevOps 很可能会淡出开发者的技术选型。

总结

最后，我来总结一下。这节课，我带你学习了 GitOps 的理论基础，例如什么是 GitOps、GitOps 的 4 个原则及其优势，以及 GitOps 为什么会成为交付标准的几大原因。

从 GitOps 工作组制定的原则我们可以得出结论，最常用的 GitOps 工具 FluxCD 和 ArgoCD 都是基于相同的理念来构建的。至于 GitOps 的优势，通过之前的实战我相信你已经感受到了它的强大之处。

此外，GitOps 之所以能成为云原生交付的事实标准，我个人认为除了文中提到的三个原因以外，还有一个非常重要的原因是云原生技术特别是容器化和 Kubernetes 技术逐渐成为了行业的标准。在这种情况下，旧的 CI/CD 工具也显现出了效率低和适配性不够强的问题，社区迫切需要更好的应用交付体验，这就为 GitOps 的普及提供了更多的可能性。

总的来说，GitOps 吸收了 DevOps 的优势，同时也借鉴了 SRE（网站可靠性）的思想，给我们带来了颠覆式的应用交付体验。

分享给需要的人，可获赠课程，你将得 18 元

生成海报并分享

赞 4　提建议

上一篇　37丨我该从哪些方向了解云原生领域？

下一篇　39丨GitOps 最佳实践，ArgoCD 凭什么脱颖而出？

精选留言 (3)

写留言

林龍

2023-03-06 来自广东

老师，不知道我的理解对不对

1.devops的做法是。我们一般提交代码到git后jenkins的做法是执行一系列命令完成部署，由于k8s是一个非常重要的组件所以不允许jenkins执行操作k8s相关的命令
2.如果把yaml文件放在了git中，由k8s去监听git的变化，把被动变成主动。同时让k8s的命令操作不允许外部来执行。
3.通过git记录可以监控/记录k8s的操作记录，让变更有据可查。

4.项目git代码与git yaml配置分离可以很好的避免让开发直接操作配置文件导致k8s的变更（自己公司遇到了有人改了jenkins流水线，导致构建失败）

作者回复: 是的，这几点都总结得非常正确。

3

Waylon

2023-03-26 来自北京

"在推模式下，CI 或者 CD 修改集群对象时，它需要在集群外部取得集群的凭据.....
而 GitOps 通过 Operator 在集群内实现了拉的模式，在解决凭据安全问题的同时....."
老师，文中提到的，如果以ArgoCD为例，ArgoCD部署在本集群的话，确实是不需要额外添加kubeconfig做认证，但是生产环境下，通常都是多个集群，如果让ArgoCD支持多集群部署，也需要用argocd命令添加kubeconfig，进而将目标集群添加到ArgoCD.
额外添加的kubernetes cluster，相对于ArgoCD 而言，也是外部集群呀，同样是通过kubeconfig获取了集群认证凭据，进而才能通过gitrepo协调吗。

所以老师，我对于您描述的上述信息，不是特别理解，希望可以再进一步答疑解惑。

作者回复: 无论是单集群还是多集群，凭据实际上都不会离开集群。只不过多集群的管理方式相当于把其他集群的凭据报错在了 Argo CD 的运行集群中。
另外多集群的方式有两种，你提到的是其中一种方式，另一种是为每一个集群部署一个 Argo 实例。

共 2 条评论>

1

夜空中最亮的星

2023-03-06 来自北京

GitOps 确实会越来越多的使用了