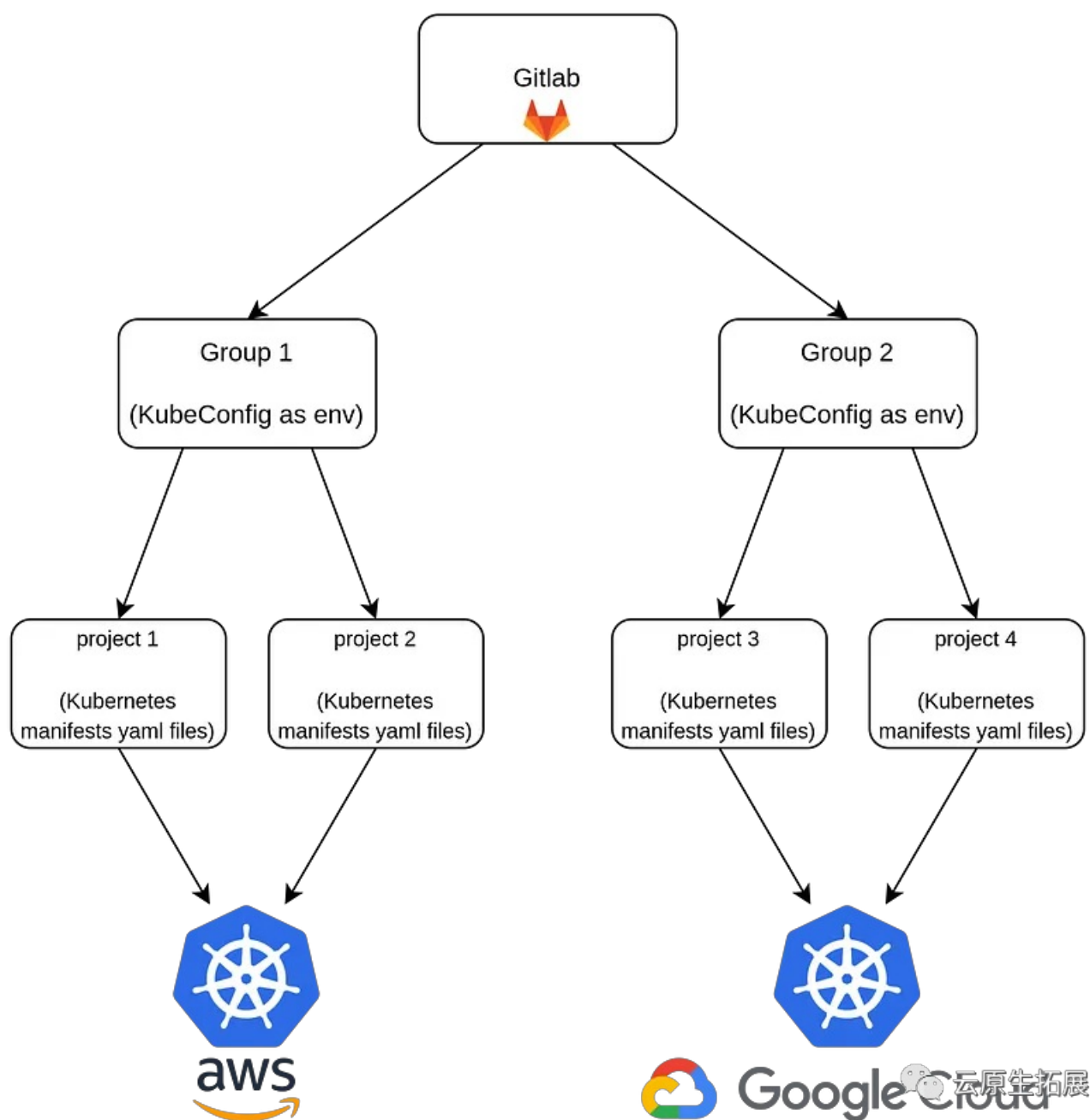


120Kubernetes 系列（一一三）使用 Git 管理多个 Kubernetes 集群

使用 GitLab 在中央统一位置管理多云 Kubernetes 集群。



介绍

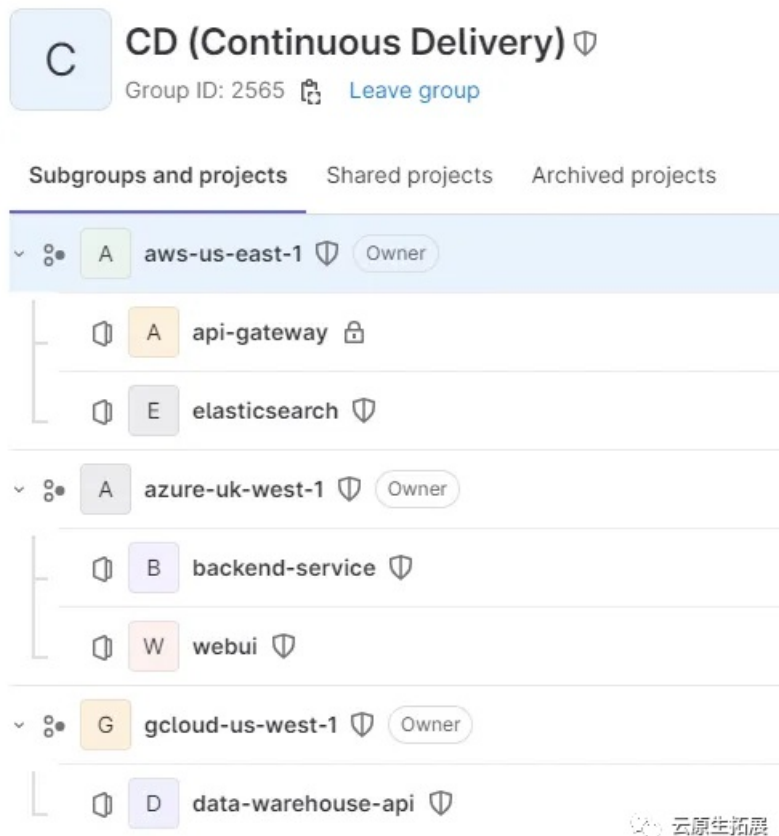
有效管理多个 Kubernetes 集群对于采用云原生架构的组织至关重要。GitLab 和 GitHub 是提供广泛功能的流行平台，但在有效管理多个 Kubernetes 集群时，GitLab 成为更好的选择。在本文中，我将探讨为什么 GitLab 在此背景下大放异彩，重点关注其简化集群管理、强化结构以及通过层次结构组增强协作的能力。我们还将深入研究使用 GitLab 强大的模板功能封装 CI/CD 管道代码的优势。

GitLab or GitHub ?

GitLab 和 GitHub 都提供了许多令人惊叹的功能，但特别是对于管理多个 Kubernetes 集群，GitLab 是更好的选择。您可以在 Git 存储库中有效管理清单，在多云环境下以结构化方式组织它们，类似于 GitOps 实践。GitLab 提供了强大的层次结构，允许您定义无限的组，而 GitHub 则施加限制，阻止定义层次组。这就是我更喜欢 GitLab 而不是 GitHub 的原因之一。

1 创建 GitLab 层次结构组

您可以在另一个组中定义一个组并继承所有父组的环境变量。在这种情况下，您可以在父组中定义 KUBECONFIG 文件，并在其中的所有项目中使用这些变量。假设我在 AWS、Google Cloud 和 Azure 等不同环境中拥有三个 Kubernetes 集群。首先，您需要创建一个组和项目层次结构，如下所示：



每个组代表一个 Kubernetes 集群，每个项目都包含一组与该项目相关的清单文件。例如，在上图中，aws-us-east-1 组中有一个名为 api-gateway 的项目。这表明该项目中的清单旨在部署到位于 us-east 区域的 Kubernetes 集群。

2 将 KubeConfig 放入组级环境变量中

GitLab 组内的项目可以访问在其父组或更高级别中设置的共享环境变量，反之亦然。为此，您需要在组设置中配置 KubeConfig 文件（包含用于连接到 Kubernetes API 服务器的凭据的配置文件）：Group Name -> CI/CD Settings -> Variables -> Add Variable 路径：

Add variable

Key: KUBECONFIG

Value:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    api-version: v1
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tck1JSUM0VENQWNTZ0F3SUJBZ0lCQURBTk3na3Foa2lHOXcwQkFRc
0ZBREFTTVjBd0RnWURWUVRERXdkcmRXS2Vkd0aE1CNFhEVEl5TVRJeE5ERXZ0RFF6TmVvWERUTXlNVE14TVRFd0
5EUXpNbG93RmpFUU1BNEdBMVVFQXhNSAphM1ZpwIMxallUQ0NB013RFFZSktvWk1odmNOQVFFQk3RQRnZ0VQUR
DQ0FRB0NnZ0VCQutxODBNs2UvNDF2CjdadKE1SXFEBzARYU01eXVYQXkzWVM4ZDZYZld4dDdHc1BwbDU2b1RkbE9U
ZDMzekxTMzRXL2hDTk4rbVVEwZjgKdMNGNDV1d0ZEOTJkTWZNMtqZTdTc3Z5T2FIRitodk91VW1JUl1drSS9ndw10b
```

Type: ENV_VAR

Environment scope: All (default)

Flags:

- ☒ Protect variable
Export variable to pipelines running on protected branches and tags only.
- ☐ Mask variable
Mask this variable in job logs if it meets regular expression requirements.
- ☒ Expand variable reference
\$ will be treated as the start of a reference to another variable.

Buttons: Cancel, Add variable

例如，在我的场景中，我需要将 Kubernetes 集群的 Kubeconfig 文件放置在各自的组中。

3 使用 GitLab CI 部署您的应用程序

现在，您可以在每个项目中创建清单文件并使用 GitLab CI 管道部署它们。以下是 aws-us-east-1 组中 api-gateway 项目的内容示例：

```
.
├── gitlab-ci.yml
└── manifests
    ├── configmap.yaml
    ├── deployment.yaml
    └── service.yaml
```

在 gitlab-ci.yml 文件内，您可以通过以下方式部署清单：

```
stages:
  - deploy

deploy:
  script:
    - kubectl apply -f manifests
```

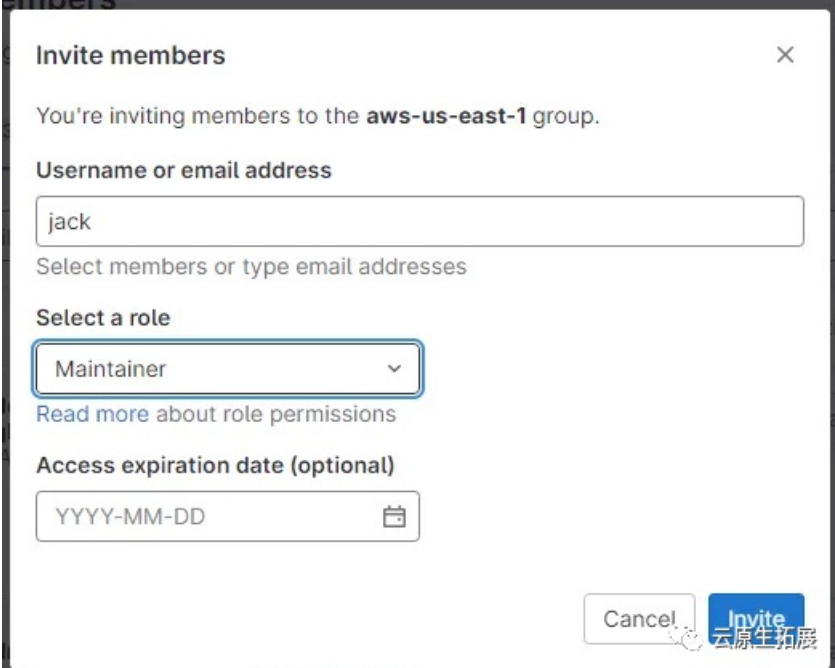
```
stages:
  - deploy

deploy:
  stage: deploy
  image: bitnami/kubectl
  script: |
    for f in manifests/*
    do
      kubectl apply -f $f
    done
```

KUBECONFIG 环境变量在父组中定义，因此无需为每个单独的项目定义它。

4 访问控制

您可以以分层方式将组和管理委派给团队成员。通过在组级别分配访问权限，该组内的所有项目都将继承相同的访问模型，从而无需单独修改每个项目的访问设置。例如，在下图中，我已授予我的团队成员之一 Jack Maintainer 访问权限（包括读取和写入权限），以修改 `aws-us-east-1` 组：



Invite members

You're inviting members to the **aws-us-east-1** group.

Username or email address

jack

Select members or type email addresses

Select a role

Maintainer

[Read more about role permissions](#)

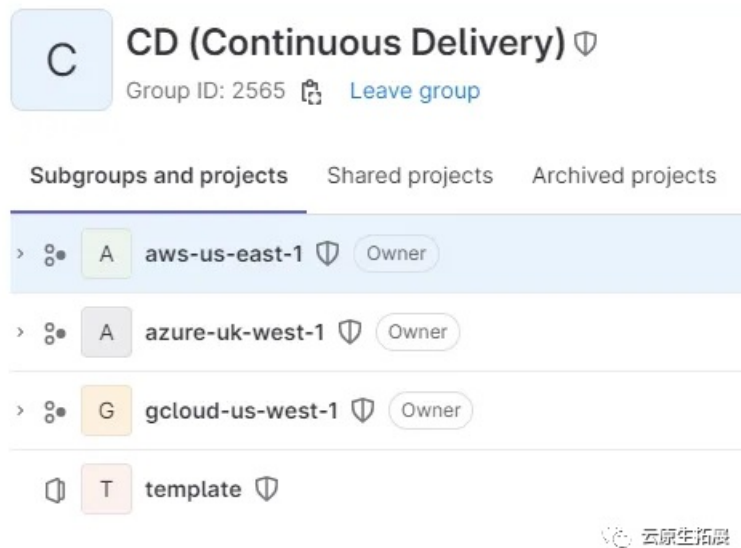
Access expiration date (optional)

YYYY-MM-DD

Cancel Invite

5 封装 CD 管道代码

GitLab 的另一个优势是它的模板功能。您可以为管道定义通用模板并将其包含在您需要的任何地方。这可以帮助您封装 CD 管道代码并在一个位置进行更改。因此，如果您需要修改管道代码，则不必单独对每个项目进行更改。首先，创建一个项目来存储您的管道代码，然后将其包含到其他项目中。在我的例子中，我将这个项目命名为 CD 组下的 `template`，如下所示：



在模板项目中，有一个简单的 `install.yml` 文件，其内容与上一节中提到的 `gitlab-ci.yml` 文件类似。

```
stages:
  - deploy

deploy:
  stage: deploy
  image: bitnami/kubectl
  script: |
    for f in manifests/*
    do
      kubectl apply -f $f
    done
```

现在，您需要更新所有项目并将此内容放入 `gitlab-ci.yml` 文件中：

```
include:
  - project: 'cd/template'
    ref: 'main'
    file: 'install.yml'
```