



容器服务 最佳实践

文档版本: 20211019



法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。 如果您阅读或使用本文档,您的阅读或使用行为将被视为对本声明全部内容的认可。

- 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档,且仅能用 于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息,您应当严格 遵守保密义务;未经阿里云事先书面同意,您不得向任何第三方披露本手册内容或 提供给任何第三方使用。
- 未经阿里云事先书面许可,任何单位、公司或个人不得擅自摘抄、翻译、复制本文 档内容的部分或全部,不得以任何方式或途径进行传播和宣传。
- 由于产品版本升级、调整或其他原因,本文档内容有可能变更。阿里云保留在没有 任何通知或者提示下对本文档的内容进行修改的权利,并在阿里云授权通道中不时 发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠 道下载、获取最新版的用户文档。
- 4. 本文档仅作为用户使用阿里云产品及服务的参考性指引,阿里云以产品及服务的"现状"、"有缺陷"和"当前功能"的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引,但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的,阿里云不承担任何法律责任。在任何情况下,阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害,包括用户使用或信赖本文档而遭受的利润损失,承担责任(即使阿里云已被告知该等损失的可能性)。
- 5. 阿里云网站上所有内容,包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计,均由阿里云和/或其关联公司依法拥有其知识产权,包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意,任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外,未经阿里云事先书面同意,任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称(包括但不限于单独为或以组合形式包含"阿里云"、"Aliyun"、"万网"等阿里云和/或其关联公司品牌,上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司)。
- 6. 如若发现本文档存在任何错误,请与阿里云取得直接联系。

通用约定

格式	说明	样例
⚠ 危险	该类警示信息将导致系统重大变更甚至故 障,或者导致人身伤害等结果。	⚠ 危险 重置操作将丢失用户配置数据。
▲ 警告	该类警示信息可能会导致系统重大变更甚 至故障,或者导致人身伤害等结果。	警告 重启操作将导致业务中断,恢复业务 时间约十分钟。
〔〕 注意	用于警示信息、补充说明等,是用户必须 了解的内容。	大) 注意 权重设置为0,该服务器不会再接受新 请求。
⑦ 说明	用于补充说明、最佳实践、窍门等,不是 用户必须了解的内容。	⑦ 说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击设置> 网络> 设置网络类型。
粗体	表示按键、菜单、页面名称等UI元素。	在 结果确认 页面,单击 确定 。
Courier字体	命令或代码。	执行 cd /d C:/window 命令,进入 Windows系统文件夹。
斜体	表示参数、变量。	bae log listinstanceid
[] 或者 [alb]	表示可选项,至多选择一个。	ipconfig [-all -t]
{} 或者 {alb}	表示必选项,至多选择一个。	switch {act ive st and}

目录

1.容器服务swarm集群与Kubernetes集群的主要功能比对	06
1.1. 概述	06
1.2. 概念比对	06
1.3. 基本配置比对(使用镜像创建应用)	09
1.4. 网络配置比对(使用镜像创建应用)	10
1.5. 数据卷及环境变量配置比对(使用镜像创建应用)	17
1.6. 容器配置及标签比对(使用镜像创建应用)	18
1.7. 健康检查及自动伸缩比对(使用镜像创建应用)	19
1.8. 使用yaml文件创建应用比对	21
1.9. 网络比对	26
1.10. 日志及监控比对	26
1.11. 应用访问比对	27
2.在阿里云容器服务上运行基于 TensorFlow 的 Alexnet	31
3.节点重启操作最佳实践	33
4.使用 OSSFS 数据卷实现 WordPress 附件共享	34
5.使用 Docker Compose 测试集群网络连通性	37
6.镜像服务	39
6.1. Docker 镜像的基本使用	39
6.2. 镜像服务主子账号授权	41
7.日志	45
7.1. 容器服务中使用 ELK	45
7.2. Docker 日志收集新方案: log-pilot	49
8.Docker 容器健康检查机制	54
9.一键部署 Docker Datacenter	57
10.在容器服务上轻松搭建 Concourse Cl	61
11.利用Terraform部署Swarm集群及Wordpress应用	68

12.Chef实现Dokcer和WebServer自动化部署	76
--------------------------------	----

1.容器服务swarm集群与Kubernetes集 群的主要功能比对

1.1. 概述

本文将介绍容器服务swarm集群与Kubernetes集群主要功能比对的前提条件及使用限制。

前提条件

您已经成功创建一个 Kubernetes 集群,参见创建Kubernetes专有版集群。

? 说明。

- 目前容器服务Kubernetes版支持四种集群:经典集群、Kubernetes托管版、多可用区 Kubernetes及Serverless Kubernetes(公测)。
- 本文以创建Kubernetes集群为例,进行容器服务Swarm集群与Kubernetes集群的功能比对。

使用限制

本文主要介绍以下两种场景的功能比对:

- 应用均为无状态应用。
- 应用的数据存在数据库或存储中。

1.2. 概念比对

本文主要介绍容器服务Swarm集群与Kubernetes集群主要概念的比对。

应用

容器服务Swarm集群

容器服务Swarm中,应用类似于项目,一个应用下面可以有多个服务。服务是具体提供应用功能的实例。服务可以水平扩展。



容器服务Kubernetes集群

容器服务Kubernetes中,应用是指部署(deployment),能够提供应用对外暴露的功能。部署中会有Pod和container,但Kubernetes中最小的调度单位是一个pod,其中Pod可包含多个container。一个Pod可以认为是应用的一个实例,多实例(多Pod)可以调度到不同的节点上,也就是说Pod可以水平扩展。



② 说明 虽然上图中一个Pod中有多个container,但是在实际使用时,建议一个Pod对应一个 container,这里对应多个container是为了说明Pod的能力。

服务

容器服务Swarm集群

容器服务Swarm中的服务即提供应用功能的具体实例。当在Swarm集群中创建一个应用的时候,服务的访问 方式会直接暴露给外部。

容器服务Kubernetes集群



容器服务Kubernetes中的服务是一个抽象概念,通过服务(Service)可以将应用(Deployment)的访问方 式暴露给外部。

应用访问

容器服务Swarm集群

容器服务Swarm在部署应用的时候,可以选择三种应用访问方式,无论哪一种方式都可以直接暴露应用访问,不需要额外的操作:

- <HostIP>:<port>
- 简单路由
- 负载均衡

容器服务Kubernetes集群

容器服务Kubernetes在创建应用(即部署deployment)后,不能直接暴露应用的访问方式,需要通过创建服务(Service)进行应用访问的暴露。在容器服务Kubernetes集群内部应用之间可以通过服务名(Service Name)进行访问,服务名也只能用于集群内部访问。若要在集群外部访问应用,需要创建NodePort类型和LoadBalancer类型的服务进行对外暴露:

- ClusterIP (集群内部访问使用,也可以使用服务名)
- NodePort(类似于Swarm集群的<HostIP>:<port>)
- LoadBalancer (类似于Swarm集群的负载均衡)
- 域名,通过创建路由(ingress)来实现(类似于Swarm集群的简单路由)

1.3. 基本配置比对(使用镜像创建应用)

本文介绍容器服务Swarm集群与Kubernetes集群使用镜像创建应用时,基本配置的比对。

使用镜像创建应用

容器服务Swarm与Kubernetes集群在使用镜像创建应用时,部署界面差异较大。

- 容器服务Swarm集群,请参见创建应用。
- 容器服务Kubernetes集群,请参见创建无状态工作负载Deployment。

应用基本信息

容器服务Swarm集群

在应用基本信息中,部署的内容包括应用名称、应用版本、部署集群、默认更新策略及应用描述。

创建应用 1 1 1 2	回应用列表				
常见问题: 🔗 限制语	緊體的资源 🔗 高可用性调度 🔗 通过镜像创建Nginv	の通过編排模板创建Wordpress の編排模板の	朝 🔗 标签说明		
	应用基本信息		应用配置	\rangle	创建完成
应用名称	nginx				
	名称为1-64个字符,可包含数字、英文字符,或"-",	且不能以-开头			
应用版本:	1.0				
部署集群:	swarm-cluster	•			
默认更新策略:	标准发布	Ŧ			
应用描述:					
		II.			
	■ 检查最新Docker镜像 ①				
					使用编集创建使用编排模板创建

容器服务Kubernetes集群

与Swarm集群不同的是:需要配置命名空间、副本数量及类型。

命名空间是容器服务Kubernetes集群特有的概念,Kubernetes通过命名空间(namespace)进行资源的隔 离,如CPU等。同时也可以区分不同的使用环境,例如测试环境、开发环境。如果涉及生产环境,建议通过 集群隔离。Kubernetes的命名空间,可参见基本概念。

创建应用								
	应用基本信息	ş	容器配置	\rangle	高级配置	\rangle	创建完成	
应用名称	nginx							
	名称为1-64个字符,可包含数字、小写英文字符	F,或"-",且不能以-开头						
部署集群:	kubernetes-test	Ŧ						
命名空间:	default	Ŧ						
副本数量:	2							
类型	无状态	¥						
							返回	F—₩

基本配置

基本配置, 主要是选择镜像和镜像版本。

容器服务Swarm集群

网络模式目前支持默认和host 两种。

	镜像名称:	nginx	选择镜像		
町	镜像版本:	latest	选择镜像版本		
理本理	容器数量:	1	网络模式:	默认	v
	Restart :	✓ Always			

容器服务Kubernetes集群

- 网络模式,在创建集群时已经选定,可选择两种网络插件Flannel和Terway,请参见使用Terway网络插件。
- 所需资源是声明需要的CPU和内存资源,资源限制是实际使用中不能超过的资源上限。类似于容器服务 Swarm集群的容器配置部分的CPU限制和内存限制。

容	書1 ○ 添加容器		
	镜像名称:	nginx	选择镜像
	镜像版本:	latest	选择镜像版本
aleri		□ 总是拉取镜像 设置镜像密钥	
基本配置	资源限制:	CPU 2 Core 内存 4096 MiB ① 清根据实际	使用情况设置
	所需资源:	CPU 1 Core 内存 1024 MiB ①请根据实际	使用情况设置
	Init Container		

1.4. 网络配置比对(使用镜像创建应用)

本文介绍容器服务Swarm集群与Kubernetes集群使用镜像创建应用时,网络配置的比对。

使用镜像创建应用

容器服务Swarm与Kubernetes集群在使用镜像创建应用时,部署界面差异较大。

• 容器服务Swarm集群,请参见创建应用。

• 容器服务Kubernetes集群,请参见创建无状态工作负载Deployment。

网络配置

容器服务Swarm集群的网络配置主要完成应用对外访问方式的暴露。

端口映射

容器服务Swarm集群

容器服务Swarm集群的端口映射是将应用端口映射到宿主机,在每个宿主机上都会启用相同的端口,这样访问应用的时候只需要 <HostIP>:<Port> 即可访问。

端口映射:	• 如何给公网的服务添加域名						
	主机端口		容器端口		协议		
	e.g. 8080	>	e.g. 8080	1	TCP	v (•
	注意:主机端口不能设置为9080,2376,3376						

容器服务Kubernetes集群

在容器服务Kubernetes集群可以通过NodePort类型的Service来实现,有以下两种方法:

方法一: 创建应用时配置

1. 部署完容器配置后,进行高级配置时,在访问设置区域,单击服务(Service)右侧的创建。

创建	应用					
		应用基本信息	\rangle	容器配置	高级配置	创建完成
揽	服务(Service)	创建				
访问说	路由(Ingress)	创建				

2. 类型请选择节点端口。具体操作,请参见创建无状态工作负载Deployment。

创建服务		\times
名称:	nginx-svc	
类型:	でであるというでは、「「」では、「」では、「」では、「」では、「」では、「」では、「」では、「	
端口映射:	●添加	
	服务端口 容器端口 节点端口 协议	
	e.g. 8080 80 30000-327 TCP 🔻 🖨	
注解:	● 添加	
标签:	○ 添加	
	创建取	肖

方法二:通过创建服务配置

1. 登录容器服务管理控制台。

- 2. 在控制台左侧导航栏中,单击集群。
- 3. 在集群列表页面中,单击目标集群名称或者目标集群右侧操作列下的详情。
- 4. 在集群管理页左侧导航栏中,选择网络 > 服务。
- 5. 选择命名空间,单击创建,在创建服务页面,类型选节点端口。具体操作,请参见管理服务。

创建服务			\times
	名称:		
	类型:	节点端□	
	关联:	▼	
赫	归映射:	●添加	
		服务端口 容器端口 节点端口 协议	
		e.g. 8080 e.g. 8080 30000-327 TCP 🔻 🗢	
	注解:	● 添加	
	标签:	◎ 添加	
		创建取消	Ц.

简单路由配置

容器服务Swarm集群

容器服务Swarm集群的简单路由配置为用户提供了通过域名访问应用的方式,用户可以选择使用容器服务 提供的域名也可以自定义域名。

简单路由配置:	O 如何暴露 HTTP 服务		
	容器端口	域名	
	e.g. 80	如 [<schema>://]<domain-name>[/<context>]</context></domain-name></schema>	•
	注意:相同端口的多个域名只能填写在同一个条目内。多个域名	用; 分隔	

容器服务Kubernetes集群

在容器服务Kubernetes集群可以通过路由(Ingress)功能来实现,用户可以通过创建路由的方式进行相关功能的创建。同时容器服务Kubernetes的Ingress还提供了蓝绿发布、灰度发布等功能。更多信息,请参见通过 Ingress实现灰度发布和蓝绿发布。

容器服务Kubernetes集群实现路由(Ingress)功能,有两种方法:

方法一: 创建应用时配置

1. 部署完容器配置后,进行高级配置时,在访问设置区域,单击路由(Ingress)右侧的创建。

创建	並用					
		应用基本信息	\rangle	容器配置	高级配置	创建完成
則	服务(Service)	创场重				
访问说	路由(Ingress)	创成量				

2. 使用镜像创建无状态应用。具体操作,请参见创建无状态工作负载Deployment。

创建		\times
名称:	nginx-ingress	
规则:	• 添加	
	域名 ● 使用 *.(或者 自定义 路径 ● e.g./ ● 服务 ● 添加 端口 ▼ ▼ ● 开启TLS	
服务权重:		
灰度发布策略:	♀ 添加 同时设置灰度规则和权重,满足灰度规则的请求将会继续依据权重路由到新老版本服务中	
注解:	● 添加 重定向注解	
标签:	• 添加	
	创建	取消

方法二:通过创建路由配置

- 1. 登录容器服务管理控制台。
- 2. 在控制台左侧导航栏中,单击集群。
- 3. 在集群列表页面中,单击目标集群名称或者目标集群右侧操作列下的详情。
- 4. 在集群管理页左侧导航栏中,选择网络>路由。
- 5. 选择命名空间,单击创建。具体操作,请参见控制台操作指导。

创建		
名称:	nginx-ingress	
规则:	● 添加	
	域名	8
	使用 *.(
	路径 e.g./	
	服务 📀 添加	
	名称	
	▼ ▼ ●	
	□ 开启TLS	
服务权重:		
灰度发布策略:	♀ 添加 同时设置灰度规则和权重,满足灰度规则的请求将会继续依据权重路由到新老版本服务中	
注解:	● 添加 重定向注解	
标签:	● 添加	
	CUB	取

负载均衡路由配置

容器服务Swarm集群

容器服务Swarm集群的负载均衡路由配置为应用提供了通过阿里云负载均衡(Server Load Balancer)暴露 应用访问方式的能力,用户首先自己创建SLB,然后将创建的SLB的IP及端口信息绑定到应用上,用户可以通 过<SLB_IP>:<Port>的方式访问应用。

负载均衡路由配置	• 心 如何使用自定义负载均衡方式暴露服务		
	容器端口	自定义负载均衡	
	e.g. 80	\$□ [http https tcp]://[slb name slb id]:[front port]	•
	注意:不同服务不能共享使用slb,不能使用该集群默认slb		

容器服务Kubernetes集群

容器服务Kubernetes集群同样支持通过绑定SLB的方式进行应用访问方式的暴露。容器服务Kubernetes集群 上创建SLB是通过LoadBalancer类型的Service进行自动创建,不需要手工创建后配置。自动创建SLB时,可 以选择公网访问或私网访问。同时如果使用YAML文件进行创建的话,还可以指定已有SLB及支持会话保持等 配置。具体操作,请参见管理服务。

容器服务Kubernetes集群上创建Loadbalancer类型的Service有两种方式:

方法一: 创建应用时配置

1. 部署完容器配置后,进行高级配置时,在访问设置区域,单击服务(Service)右侧的创建。

创建应用							
	应用基本信息	\geq	容器配置	高级配置	>	创建完成	
服务(Service	() 创建						
置 露曲(Ingress) 创建						

2. 类型请选择负载均衡。具体操作,请参见创建无状态工作负载Deployment。

创建服务			\times
	名称:	nginx-svc	
	类型:	负载均衡 ▼ 公网访问 ▼	
羢	□映射:	 ◆添加 服务端口 容器端口 协议 e.g. 8080 80 TCP ▼ ● 	
	注解:	 ○ 添加 负载均衡配置参数 	
	标签:	☑ 添加	
		创建取消	-

方法二:通过创建服务配置

- 1. 登录容器服务管理控制台。
- 2. 在控制台左侧导航栏中,单击集群。
- 3. 在集群列表页面中, 单击目标集群名称或者目标集群右侧操作列下的详情。
- 4. 在集群管理页左侧导航栏中,选择网络 > 服务。
- 5. 选择命名空间,单击创建,在创建服务页面,类型选负载均衡。更多信息,请参见管理服务。

创建服务			\times
	名称:		
	类型:	负载均衡 ▼ 公网访问 ▼	
	关联:	v	
놽	 山映射:	 ●添加 服务端口 容器端口 协议 e.g. 8080 e.g. 8080 TCP ▼ ● 	
	注解:	● 添加 负载均衡配置参数	
	标签:	● 添加	
		创建取	消

1.5. 数据卷及环境变量配置比对(使用镜像创建 应用)

本文介绍容器服务Swarm集群与Kubernetes集群使用镜像创建应用时,数据卷及环境变量的配置比对。

使用镜像创建应用

容器服务Swarm与Kubernetes集群在使用镜像创建应用时,部署界面差异较大。

- 容器服务Swarm集群,请参见创建应用。
- 容器服务Kubernetes集群,请参见创建无状态工作负载Deployment。

数据卷

容器服务Swarm集群

填写用户申请的云存储或者本地存储路径。

	数据卷:	✤ ௴如何使用第三方数据卷				
		主机路径或数据卷名	容器路径		权限	
数据卷				/	RW	•
	volumes_from :					

容器服务Kubernetes集群

容器服务Kubernetes集群使用存储的方式与容器服务Swarm集群一样,只是挂载的过程不一样,控制台界面 基本一致。

数据卷:	😌 増加本地存储	加本地存储 博士 建築 春器路径 諸小目录 諸協入主机路径、如/tmp 诸協入主航容器路径、如/tmp 御広存储 王載源 審器路径 辞学型 注載源 年調源 至需器路径 文 请选择 文 请输入主载容器路径、如/tmp		
	存储卷类型	挂载源	容器路径	
	主机目录	✔ 请输入主机路径,如/tmp	请输入挂载容器路径,如/tmp	•
数据卷	增加云存储			
	存储卷类型	挂载源	客謡路径	
	云存储	✓ 请选择	请输入挂载容器路径,如/tmp	۰

用户可以选择本地存储,也可以选择云存储:

- 本地存储:包括主机目录、Kubernetes集群特有的配置项(configmap)、保密字典(secret)及临时目录。
- 云存储:包括云盘、NAS及OSS。

环境变量

容器服务Swarm集群与容器服务Kubernetes集群环境变量的配置一样。用户只需输入键值即可。

	环境变量:	● 新増			
遼重		类型	变量名称	查量/查量引用	
环境		自定义 🔻	e.g. foo	e.g. foo	•

1.6. 容器配置及标签比对(使用镜像创建应用)

本文介绍容器服务Swarm集群与Kubernetes集群使用镜像创建应用时,容器配置及标签的比对。

使用镜像创建应用

容器服务Swarm与Kubernetes集群在使用镜像创建应用时,部署界面差异较大。

- 容器服务Swarm集群,请参见创建应用。
- 容器服务Kubernetes集群,请参见创建无状态工作负载Deployment。

容器配置

容器服务Swarm集群

设置容器的启动命令(Command和Entrypoint)、资源限制(CPU限制和内存限制)及启动项等配置。

	Command :						
	Entrypoint :						
問	CPU限制:				内存限制:	МВ	
容器配	Capabilities:	ADD	DROP				
	容器启动项:	🗖 stdin 🗖 tty					
	HostName:						

容器服务Kubernetes集群

容器服务swarm集群的容器配置,类似于容器服务Kubernetes集群的生命周期配置和基本配置。

- 生命周期,详情请参见创建无状态工作负载Deployment。
 - 启动执行
 - 启动后处理
 - 停止前处理

	容器启动项:	Stdin 🗎 tty
	启动执行:	命令
開開		参数
#1	启动后处理:	命令
	停止前处理:	命令

- 基本配置, 详情请参见创建无状态工作负载Deployment, 推荐配置请参见高可靠推荐配置。
 - 资源限制
 - 所需资源

标签

容器服务Swarm集群:能够完成健康检查、设置访问域名、日志等功能。

容器服务Kubernetes集群:容器服务Kubernetes集群中的标签只能标识一个应用。容器服务Kubernetes部 署的应用在创建时,会自动生成与应用名相同的标签,在使用镜像创建应用的配置界面没有展现,用户可以 通过yaml文件的方式使用此标签。

1.7. 健康检查及自动伸缩比对(使用镜像创建应 用)

本文介绍容器服务Swarm集群与Kubernetes集群使用镜像创建应用时,健康检查及自动伸缩的比对。

使用镜像创建应用

容器服务Swarm与Kubernetes集群在使用镜像创建应用时,部署界面差异较大。

- 容器服务Swarm集群,请参见创建应用。
- 容器服务Kubernetes集群,请参见创建无状态工作负载Deployment。

健康检查

● 容器服务Swarm集群

健康检查是通过标签的方式实现。

● 容器服务Kubernetes集群

在使用镜像创建应用的容器配置页签,可进行健康检查的配置。目前支持存活检查和就绪检查。

存活检查	- /					
		Http请求	TCP连接		命令行	~
	that/					
	B24X	HTTP		*		
	路径					
	山影					
	Http头	name	value			
	延迟探测时间(秒)	3				
	执行探测频率(秒)	10				
	超时时间(秒)	1				
	健康阈值	1				
	不健康阈值	3				
就绪检查	🔲 开启					
就绪检查存活检查	 开启 开启 					
就維检查 存活检查	 开启 一开启 					
就绪检查 存活检查 就绪检查	 开启 开启 ✓ 开启 					
就绪检查 存活检查 就绪检查	 开启 开启 【 【 开启 	Http請求	TCP连接		命令行	
就绪检查 存活检查 就编检查	 开启 开启 了开启 (1) (1)	Http请求	TCP连接		命令行	
就須检查 存活检查 就绪检查	 开启 开启 ✓ 开启 ////////////////////////////////////	Http 游求 HTTP	TCP连接		命令行	
就油检查 存活检查 就结检查	 开启 开启 グ 开启 协议 路径 	Нітр訴求 НТТР	TCP连接		命令行	
就编检查 存/活检查 就编检查	 开启 开启 ブ用店 が议 路径 端口 	Нцр調求 НТТР	TCP连接		命令行	
就街检查 存活检 <u>章</u> 就绪检查	 开启 开启 了开启 が议 路径 端口 Http头 	Http游求 HTTP name	TCP连接	•	命令行	
就绪检查 存活检查 就缘检查	 开启 开启 / 开启 / 小店 /	Http://www.htttp://www.http://www.http://www.http://www.http://www.http://www.http://www.http://www.http://www.http://www.http://www.htttp://www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.htttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttp.//www.httttttp.//www.httttttp.//www.htttttttttp.//www.htttttttttttttttttttttttttttttttt	TCP连接 value		命令行	
就绪检查 存活检查 就绪检查	 开启 开启 (一) 开启 (1) 开启 (1) 折応 (1) 話径 (1) 請口 (1) Http共 (1) 延迟探测时间(秒) (1) 折沉探测频率(秒) 	Http 游求 HTTP name 3 10	TCP连接		命令行	
就插检查 存活检查 就结检查	 开启 开启 了开启 が议 路径 端口 Http头 延迟探測时间(秒) 执行探测频率(秒) 超时时间(秒) 	Http請求 HTTP name 3 10	TCP连接		命令行	
就绪检查 存活检查 就缘检查	 开启 开启 一开启 使用 使用 防议 路径 靖口 Http头 延迟採測时间(秒) 执行採測版承報(秒) 超即时间(秒) 磁測面值 	Http请示 HTTP name 3 10 1	TCP连接		命令行	
就编检查 存/活检查 就/编检查	 开启 开启 开启 が以 路径 端口 Http头 延迟探測时间(秒) 执行探测频率(秒) 基时时间(秒) 磁測面 石(中国)通信 石(中国)通信 	Http://www.main.com/www.com/	TCP连接		命令行	

自动伸缩

● 容器服务Swarm集群

提供基于CPU和内存两个维度的自动伸缩。

● 容器服务Kubernetes集群

在使用镜像创建应用的**高级配置**页签,可配置容器组的水平伸缩,同样提供CPU和内存两个维度的自动伸缩。

	容器组水平伸缩	☑ 开启		
		指标: CPU使用量		•
单编记 置		触发条件:使用量 70	9/0	
and a second		最大副本数: 10	可选范围:2-100	
		最小副本数: 1	可选范围:1-100	

1.8. 使用yaml文件创建应用比对

本文介绍容器服务Swarm集群与Kubernetes集群使用yaml文件创建应用时,Swarm集群下的yaml文件与Kubernetes集群下的yaml文件的对应关系。

背景信息

在使用yaml文件创建应用时, Swarm集群与Kubernetes集群的yaml文件格式不一样:

• 您可以通过Kompose工具对Swarm集群的yaml文件进行自动转换。但转换后的内容仍需您核对检查。

Kompose工具可以从Github上获取。

Kompose工具下载地址:

- Mac下载地址: Mac
- Linux下载地址: Linux
- Window下载地址: Window

⑦ 说明 目前Kompose工具对阿里云的一些定制标签还不支持,如Kompose工具不支持的标签所示,阿里云容器服务团队会持续研发逐步覆盖各个标签的支持。

Kompose工具不支持的标签

标签	参考
external	external
dns_options	dns_options
oom_kill_disable	oom_kill_disable
affinity:service	服务部署约束(affinity:service)

• 您也可以手动改写yaml文件。

本文将基于容器服务Swarm的yaml文件,介绍Kubernetes的yaml文件如何与之对应。文章中的yaml示例, 仅作为示例,具体部署请依据具体情况添加及修改相关内容。

容器服务Swarm与Kubernetes集群的yaml文件比对

容器服务Swarm集群

容器服务Swarm集群的yaml文件*wordpress-swarm.yam*如下,注释中的阿拉伯数字与容器服务Kubernetes 集群的yaml文件的注释对应。

```
#----1
web:
image: registry.aliyuncs.com/acs-sample/wordpress:4.5
                                                     #---2
ports:
       #---3
 - '80'
environment: #---4
 WORDPRESS_AUTH_KEY: changeme
                                      #---5
 WORDPRESS_SECURE_AUTH_KEY: changeme #---5
 WORDPRESS_LOGGED_IN_KEY: changeme #---5
 WORDPRESS_NONCE_KEY: changeme
                                       #---5
 WORDPRESS_AUTH_SALT: changeme
                                       #---5
 WORDPRESS_SECURE_AUTH_SALT: changeme #---5
 WORDPRESS_LOGGED_IN_SALT: changeme #---5
 WORDPRESS_NONCE_SALT: changeme
                                        #---5
                                      #---5
 WORDPRESS_NONCE_AA: changeme
restart: always #---6
links:
          #---7
 - 'db:mysgl'
          #---8
labels:
 aliyun.logs: /var/log #---9
 aliyun.probe.url: http://container/license.txt #---10
 aliyun.probe.initial_delay_seconds: '10'
                                         #---10
 aliyun.routing.port_80: http://wordpress
                                          #---11
 aliyun.scale: '3'
                         #---12
db:
       #---1
image: registry.aliyuncs.com/acs-sample/mysql:5.7 #---2
environment: #---4
 MYSQL_ROOT_PASSWORD: password
                                      #---5
restart: always #---6
labels:
         #---8
 aliyun.logs: /var/log/mysql #---9
```

容器服务Kubernetes集群

通过容器服务Swarm集群的*wordpress-swarm.yaml*文件部署的wordpress应用,在容器服务Kubernetes集 群中对应2个服务:web和db。

在容器服务Kubernetes集群上需要2个部署(deployment)和2个服务(service)。2个Deployment创建2 个Service,2个服务分别暴露2个应用的访问方式。

容器服务Swarm集群中的web应用对应Kubernetes集群的deployment和service如下:

```
⑦ 说明 以下yaml文件的内容仅作为示例说明与容器服务Swarm集群wordpress-swarm.yam的对应 关系,不可用作实际部署。
```

• wordpress-kubernetes-web-deployment.yaml内容如下:

```
apiVersion: apps/v1 # api版本
kind: Deployment # 创建资源的类型
metadata:
name: wordpress #---1
labels: #---8 在这里的label只能做标识作用
```

容器服务

```
app. worapiess
spec: #资源创建详细内容
replicas: 2 #---12 设定实例(副本)个数
selector:
 matchLabels:
 app: wordpress
  tier: frontend
strategy:
type: Recreate
template: #模板定义POD的详细信息
 metadata:
  labels: #与前面保持一致
  app: wordpress
  tier: frontend
 spec: #定义pod中container的详细信息
  containers: #
  - image: wordpress:4 #---2 对应于镜像及版本
  name: wordpress
  env: #---4 环境变量设置,kubernetes上configmap,secret都可以通过env的方式使用。
  - name: WORDPRESS_DB_HOST
   value: wordpress-mysql #---7 通过名称指向需要访问的mysql,该名称与mysql service的名称相对应。
  - name: WORDPRESS_DB_PASSWORD #---5 密码在这里使用,但kubernetes提供了secret进行密码封装。
   valueFrom:
    secretKeyRef:
     name: mysql-pass
     key: password-wordpress
  ports: #---3 容器内应用暴露的port
  - containerPort: 80
   name: wordpress
livenessProbe: #add health check ---10 健康检查
   httpGet:
    path:/
    port: 8080
   initialDelaySeconds: 30
   timeoutSeconds: 5
   periodSeconds: 5
  readinessProbe: #add health check ---10 健康检查
   httpGet:
    path:/
    port: 8080
   initialDelaySeconds: 5
   timeoutSeconds: 1
   periodSeconds: 5
  volumeMounts: #使用存储卷,将存储卷真正挂到容器内部。
  - name: wordpress-pvc
   mountPath: /var/www/html
  volumes: #获取存储卷,需先进行PV和PVC的创建。
  - name: wordpress-pvc
  persistentVolumeClaim:
   claimName: wordpress-pv-claim
```

• wordpress-kubernetes-web-service.yaml内容如下:

apiVersion: v1 #版本号 kind: Service #创建资源类型,在这里为service。 metadata: name: wordpress labels: app: wordpress spec: ports: - port: 80 #服务端口号 selector: #通过label进行应用的关联 app: wordpress tier: frontend type: LoadBalancer #---11 定义访问方式,此处为LoadBalancer类型的service, 会自动创建SLB。

容器服务Swarm集群中的db应用对应Kubernetes集群的deployment和service如下:

⑦ 说明 以下yaml文件的内容仅作为示例说明与容器服务Swarm集群*wordpress-swarm.yaml*的对应 关系,不可用作实际部署。

• wordpress-kubernetes-db-deployment.yaml内容如下:

apiVersion: apps/v1 kind: Deployment metadata: name: wordpress-mysql labels: app: wordpress spec: selector: matchLabels: app: wordpress tier: mysql strategy: type: Recreate template: metadata: labels: app: wordpress tier: mysql spec: containers: - image: mysql:5.6 name: mysql env: - name: MYSQL_ROOT_PASSWORD valueFrom: secretKeyRef: name: mysql-pass key: password-mysql ports: - containerPort: 3306 name: mysql volumeMounts: name: wordpress-mysql-pvc mountPath: /var/lib/mysql volumes: - name: wordpress-mysql-pvc persistentVolumeClaim: claimName: wordpress-mysql-pv-claim

• wordpress-kubernetes-db-service.yaml内容如下:

apiVersion: v1 kind: Service metadata: name: wordpress-mysql labels: app: wordpress spec: ports: - port: 3306 selector: app: wordpress tier: mysql clusterIP: None

1.9. 网络比对

本文介绍容器服务Swarm集群与Kubernetes集群的网络比对。

容器服务Swarm集群

容器服务Swarm集群使用的网络有两种:

- VPC网络
- 经典网络

容器服务Kubernetes集群

容器服务Kubernetes集群的网络为VPC网络,可参见Kubernetes集群网络规划。

- 若容器服务Swarm集群使用的是VPC网络,在创建Kubernetes集群时,请选择与Swarm集群相同的VPC网 络,以便Kubernetes集群与Swarm集群的网络是联通的。
- 若容器服务Swarm集群使用的是经典网络,在创建Kubernetes集群时,由于Kubernetes集群只能为VPC网络,请先打通Swarm集群与Kubernetes集群的网络,可参见迁移方案概述。

Swarm集群与Kubernetes集群网络联通后,如果Swarm集群下存在OSS、NAS或RDS等存储产品或数据 库,会获得VPC网络可访问的IP地址,即在Kubernetes集群的VPC网络中可通过此IP地址对Swarm集群的存 储产品或数据库进行访问。

1.10. 日志及监控比对

本文介绍容器服务Swarm集群与Kubernetes集群的日志及监控功能的比对。

日志

容器服务Swarm集群:通过标签实现日志等功能。

容器服务Kubernetes集群:Kubernetes集群的日志功能,可在以下情况配置及使用。

创建Kubernetes集群:

在**创建Kubernetes集群**页面,选中**使用日志服务**后,将会在集群中自动配置日志服务插件。您可以使用已有的Project,也可以创建新的Project。

日志服务	✔ 使用日志服务	
	使用已有 Project	创建新 Project
	将自动创建名称为 k8s-	log-{ClusterID} 的 Project

具体操作,请参见组件配置。

集群创建成功后,也可以手动安装日志服务组件,请参见步骤一:启用日志服务组件Logtail。

- 创建应用时配置日志服务,请参见步骤二:创建应用时配置日志服务。
- 创建应用后使用日志服务,请参见通过DaemonSet-控制台方式采集Kubernetes文件及通过DaemonSet-控制台方式采集Kubernetes标准输出。

监控

容器服务Swarm集群和Kubernetes集群,均在**创建集群**页面,选中**在ECS节点上安装云监控插件**,即可在 云监控管理控制台查看所创建ECS实例的监控信息。具体操作,请参见组件配置。

容器服务Kubernetes集群与云监控的集成使用,请参见基础资源监控。

1.11. 应用访问比对

本文介绍容器服务Swarm集群与Kubernetes集群的应用访问比对,包括集群内部应用间访问及从集群外部访问应用。

集群内部应用间访问

容器服务Swarm集群

集群内部可以通过 links 标签,将需要被访问的服务名称设置到容器的环境变量中。

例如:使用yaml文件创建应用比对中,wordpress应用的web服务与*mysq*关联,在容器启动后,通过*mysq*这个服务名称即可完成服务的访问。

```
links: #---7
- 'db:mysql'
```

容器服务Kubernetes集群

在容器服务Kubernetes集群内部,可通过Service的ClusterIP或服务名称进行应用间访问。推荐使用服务名称 进行应用间的访问。

在创建应用时,可以将需要被访问服务的服务名称以环境变量的方式使用。

例如:使用yaml文件创建应用比对中,wordpress在调用*mysql*服务时,就是通过环境变量的方式实现。

spec:

contai	ners:		
		-	

image: wordpress:4
 name: wordpress

env:

- name: WORDPRESS_DB_HOST
- value: wordpress-mysql #---7 通过名称指向需要访问的mysql,该名称与mysql service的名称相对应。
- name: WORDPRESS_DB_PASSWORD

从集群外部访问应用

通过域名访问应用

- ? 说明
 - 无论经典网络还是VPC, 请确保网络的连通。
 - DNS具有负载均衡的能力,可将流量分发到不同的后端IP。
 - 通过域名作为应用的访问入口,可以不停机地将业务从Swarm集群迁移到Kubernetes集群。

简单路由(域名绑定到容器服务Swarm集群默认的SLB上)

为实现应用从容器服务Swarm集群迁移到Kubernetes集群,需要先在容器服务Kubernetes集群上创建应用,测试可用后,再进行应用迁移。



迁移方法

> 文档版本: 20211019

- 在容器服务Kubernetes集群上通过以下步骤创建应用:
 - i. 在容器服务Kubernetes集群上创建与Swarm集群相同类型的应用。
 - ii. 在容器服务Kubernetes集群上为应用创建Loadbalancer类型的服务(Service)。
 - iii. Loadbalancer类型的服务会创建SLB,在本例中为2.2.2.2。
 - iv. 在DNS中test.com域名的后端IP增加2.2.2.2这个IP地址。
- 验证Kubernetes集群的应用可用

通过IP地址2.2.2.2可以正常访问应用,说明容器服务Kubernetes集群上的应用可用。

• 应用迁移

将DNS中test.com域名的后端IP地址1.1.1.1删除。

后续流量经过DNS后全部流向Kubernetes集群上的应用。

简单路由(域名绑定到应用在容器服务Swarm集群上自建的SLB)

域名绑定到容器服务Swarm集群上自建的SLB与绑定到默认的SLB的区别是:

- SLB不是集群默认的, 而是自建的。
- DNS默认为云解析DNS,如果用户使用自己的域名,需要自行解析。

迁移方法

与域名绑定到容器服务Swarm集群默认的SLB的方法相同:在容器服务Kubernetes集群上创建应用,测试可用后,再进行应用迁移。



通过<Host IP>:<port>访问应用

如果您通过<Host IP>:<port>的方式访问应用,那么在原容器服务Swarm集群的基础上无法做到不停机迁移 业务,请选择业务量小的时候进行业务的迁移。

迁移方法

- 1. 在容器服务Kubernetes集群上创建应用,并通过NodePort类型的服务完成应用对外访问方式的暴露, 请参见网络配置比对(使用镜像创建应用)。
- 2. 记录该<NodePort>,将Swarm集群的<port>替换为Kubernetes集群的<NodePort>。

⑦ 说明 此步骤需要逐个停止并修改应用实例。

- 3. 将Kubernetes集群内的worker节点, 挂载到Swarm集群的SLB下。
- 4. 当有流量时,会有部分流量进入Kubernetes集群的应用,待Kubernetes集群的应用验证可用后,将SLB 中Swarm集群的节点删除,完成集群的迁移。

通过负载均衡访问应用

如果您通过负载均衡的方式访问应用,那么在原容器服务Swarm集群的基础上无法做到不停机迁移业务,请选择业务量小的时候进行业务的迁移。

迁移方法

容器Kubernetes集群LoadBalancer的使用方法与容器服务Swarm的负载均衡一样,请参见网络配置比对(使用 镜像创建应用)。

2.在阿里云容器服务上运行基于 TensorFlow 的 Alexnet

AlexNet 是 2012 年由 Alex Krizhevsky 使用五层卷积、三层完全连接层开发的 CNN 网络,并赢得了 ImageNet 竞赛(ILSVRC)。AlexNet 证明了 CNN 在分类问题上的有效性(15.3% 错误率),而此前的图片 识别错误率高达 25%。这一网络的出现对于计算机视觉在深度学习上的应用具有里程碑意义。

AlexNet 也是深度学习框架常用的性能指标工具, TensorFlow 就提供的 alexnet_benchmark.py 可以测试 GPU 和 CPU 上的性能。本文档以 AlexNet 为例,向您展示如何在阿里云容器服务上简单快速地运行 GPU 应用。

前提条件

需要基于GPU服务器的Swarm集群,本例中使用GPU计算型 gn5i的 ECS 服务器。

操作步骤

- 1. 登录容器服务管理控制台。
- 2. 在Swarm菜单下,单击左侧导航栏中的应用,然后单击页面右上角的创建应用。

容器服务	应用列表	刷新 创建应用
Kubernetes Swarm	常见问题: ② 如何始建应用 ③ 支更应用配置 ③ 简单路由监姆发布策略 ⑤ 容器器性性缔缩	2
私式	集群: EGS-duster ▼ 図 除藏系统应用 □ 除藏系统应用 □ 除藏石线应用	名称 * Q *
服务	应用名称 描述 状态 容器状态 创建时间 ▲ 更新时间 ▲	提作
集群		

3. 输入应用名称(本示例中为 alexNet)并选择集群,单击下一步。

	应用基本信息		应用配置	X	创建完成
应用名称	alexNet				
	名称为1-64个字符,可包含数字、英文字符,或	'-",且不能以-开头的提示			
应用版本:	1.0				
部署集群:	EGS-cluster	¥			
默认更新策略:	标准发布	٣			
应用描述:					
		11			
	☑ 检查最新Docker镜像				
					下一步 返回

- 4. 配置应用。
 - i. 在镜像名称栏输入 registry.cn-beijing.aliyuncs.com/tensorflow-samples/alexnet_benchmark:1.0.0-de vel-gpu 。

镜像名称:	registry.cn-beijing.aliyuncs.com/tensorflow-samples/alexn	镜像版本:	
	选择镜像		选择镜像版本

ii. 在容器配置中,填写运行的命令行,比如 python /alexnet_benchmark.py --batch_size 128 --num_b atches 100 。

	Command :	python /alexne	t_benchmark.pył			
	Entrypoint :					
器配置	CPU限制:			内存限制:	МВ	
Kát	Capabilites:	ADD	DROP			
	容器启动项:	🗆 stdin 🗆 tty				

iii. 在标签中,填写阿里云 gpu 标签,标签名为 aliyun.gpu ,标签值为调度的 GPU 数量,本示例中 为 1 。

	labels :	● 「 阿里云扩展能力的标签		
\$ (1		标签名	标签值	
臣		aliyun.gpu	1	•

5. 完成应用配置后, 单击创建创建应用。

您可以在应用列表页面,查看创建的 alexNet 应用。

应用列表					局新 创建应用
小助手:如何	创建应用 变更应用配置	简单路由蓝绿发布策略 容	器弹性伸缩		
集群: EGS-clus	ster 🔹 🗷 隐藏系统应用	□ 隐藏离线应用 □ 隐藏在	线应用		名称 •
应用名称	描述 状态	容器状态	创建时间 🔺	更新时间 ▲	操作
alexNet	● 就绪	就绪:0 停止:0	2017-03-14 12:02:03	2017-03-14 12:02:03	停止 変更配置 删除 重新部署 事件

这样您就可以在管理控制台,直接通过容器日志服务查看 AlexNet 在 GPU 集群上的性能。

操作路径:在应用列表页面,单击应用名称alexNet,在应用详情页面,单击日志。

应用:alexnet				刷新
基本信息				
应用名称: alexnet	创建时间: 2018-06-13	更新时间: 2018-06-13	所在集群: EGS-cluster	
触发器 1.每种类型的被发器只能创建1个⊘				创建触发器
目前没有任何触发器,点击右上角按钮创建触发器				
服务列表 容器列表 日志 事件 路由列表				
每个容器查看条目: 100条 ▼			按日志起始时间筛选:	下载日志
alexnet_alexnet_1 2018-06-13703:56:14.6857745112 alexnet_alexnet_1 2018-06-13703:56:14.6857708112 alexnet_alexnet_1 2018-06-13703:56:14.6857811502 alexnet_alexnet_1 2018-06-13703:56:14.685781502 alexnet_alexnet_1 2018-06-13703:56:14.6857808772 alexnet_alexnet_1 2018-06-13703:56:14.68587842 alexnet_alexnet_1 2018-06-13703:56:14.68583442 alexnet_alexnet_1 2018-06-13703:56:14.6858408272 alexnet_alexnet_1 2018-06-13703:56:14.6858408272 alexnet_alexnet_1 2018-06-13703:56:14.6858408272 alexnet_alexnet_1 2018-06-13703:56:14.6858408272 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922 alexnet_alexnet_1 2018-06-13703:56:14.6858406922	conv3 [128, 13, 13, 384] conv4 [128, 13, 13, 256] conv5 [128, 13, 13, 256] 2018-06-13 03:55:56.867219: step 1 2018-06-13 03:55:57.267805: step 2 2018-06-13 03:55:57.267805(step 2 2018-06-13 03:55:58.1266(step 4 2018-06-13 03:55:58.370634: step 1 2018-06-13 03:55:59.37065(step 1 2018-06-13 03:55:59.39065(step 2 2018-06-13 03:55:59.39062(step 2 2018-06-13 03:55:59.39062(step 2 2018-06-13 03:55:59.39062(step 2 2018-06-13 03:55:09.20062(step 2 2018-06-13 03:56:00.2012(step 2 2018-06-13 03:56:00.2012(step 2)	00, duration = 0.042 10, duration = 0.042 11 darcoss 100 steps, 0.042 +/- 0.004	sec / batch	

3.节点重启操作最佳实践

直接重启节点可能会导致集群出现异常。本文结合阿里云历史案例经验,说明了在对容器服务进行主动运维 等场景下,需要重启节点时的操作最佳实践。

检查业务高可用配置

在重启容器服务节点前,建议先检查或修正如下业务配置,以避免节点重启触发单点异常,进而导致业务可 用性受损。

• 配置数据持久化策略

建议为日志、业务配置等重要数据配置的外部卷进行数据持久化,以避免容器重建后,原有容器被删除引发数据丢失。

关于容器服务数据卷的使用,参见数据卷管理。

• 配置重启策略

建议为相应业务服务配置 restart: always 自重启策略,以便节点重启后,相应容器能自动拉起。

• 配置高可用策略

建议结合产品架构,为相应业务配置可用区调度(availability:az 属性)、指定节点调度(affinity、constraint 属性)和指定多节点调度(constraint 属性)等亲和性、互斥性策略,以避免由于相应节点重启引发单点异常。比如,对于数据库业务,建议主备或多实例部署,然后结合前述特性,确保不同实例落在不同节点上,并且相关节点不会同时重启。

操作最佳实践

建议首先参阅前述说明,检查业务高可用性配置。然后 在每个节点上(切忌同时对多个节点进行操作),依次按如下步骤操作:

1. 快照备份

建议先对节点所有关联磁盘创建最新快照进行备份,以避免由于长时间未重启服务器,导致节点关机后 启动过程中出现异常导致业务可用性受损。

2. 验证业务容器配置有效性

对于 swarm 集群,重启节点上的相应业务容器,确保容器能正常被重新拉起。

3. 验证 Docker Engine 运行有效性

尝试重启 Docker daemon,确保 Docker enginge 能正常重新启动。

4. 执行相关运维操作

执行计划内的相关运维操作,比如业务代码更新、系统补丁安装、系统配置调整等。

5. 重启节点

在控制台或系统内部,正常重启节点。

6. 重启后状态检查

重启完节点后,到 容器服务管理控制台,检查节点健康状态,检查业务容器运行状态。

4.使用 OSSFS 数据卷实现 WordPress 附件共享

本文档介绍如何通过在阿里云容器服务上创建 OSSFS 数据卷来实现 WordPress 的附件在不同容器之间的共 享。

场景

Docker 容器的兴起使得 WordPress 的部署变得很简单。通过 阿里云容器服务, 您可以使用编排模板一键部 署 WordPress。

⑦ 说明 有关使用阿里云容器服务创建 WordPress 应用的详细信息 / 参见 通过编排模板创建 WordPress。

本示例使用以下编排模板创建一个名为 wordpress 的应用。

web: image: registry.aliyuncs.com/acs-sample/wordpress:4.3 ports: - '80' environment: WORDPRESS_AUTH_KEY: changeme WORDPRESS_SECURE_AUTH_KEY: changeme WORDPRESS_LOGGED_IN_KEY: changeme WORDPRESS_NONCE_KEY: changeme WORDPRESS_AUTH_SALT: changeme WORDPRESS_SECURE_AUTH_SALT: changeme WORDPRESS_LOGGED_IN_SALT: changeme WORDPRESS_NONCE_SALT: changeme WORDPRESS_NONCE_AA: changeme restart: always links: - 'db:mysal' labels: aliyun.logs: /var/log aliyun.probe.url: http://container/license.txt aliyun.probe.initial_delay_seconds: '10' aliyun.routing.port_80: http://wordpress alivun.scale: '3' db: image: registry.aliyuncs.com/acs-sample/mysql:5.7 environment: MYSQL_ROOT_PASSWORD: password restart: always labels: aliyun.logs: /var/log/mysql

该应用包含一个 MySQL 容器和三个 WordPress 容器(aliyun.scale: '3' 是阿里云容器服务的扩展标签,指 定容器的数量。有关阿里云容器服务支持的标签,参见 标签说明)。WordPress 容器通过 link 访问 MySQL。 通过定义 aliyun.routing.port_80: http://wordpress 参见 简单路由(支持HTTP/HTTPS))。

本示例部署简单,功能齐全,但其实存在一个致命的缺陷。WordPress 上传的附件是保存在本地磁盘上的, 不同容器之间不能共享。当请求被分配到其它容器时,附件就打不开了。

解决方案

本文档介绍如何利用阿里云容器服务的 OSSFS 数据卷(OSSFS volume),无需改动任何代码,即可实现 WordPress 附件在不同容器之间的共享。

OSSFS 数据卷是阿里云容器服务提供的第三方数据卷,通过将各种云存储(比如 OSS)包装成数据卷,直接 挂载在容器上。不同容器间可以共享数据卷,并在容器重启、迁移时自动重新挂载数据卷。

操作流程

- 1. 创建 OSSFS 数据卷。
 - i. 在 容器服务管理控制台, 单击左侧导航栏中的数据卷, 即可使用数据卷功能。
 - ii. 选择需要创建数据卷的集群并单击右上角的创建,按照提示创建 OSSFS 数据卷。

有关如何创建 OSSFS 数据卷的详细信息,参见 创建 OSSFS 数据卷。

本示例中创建的 OSSFS 数据卷名称为wp_upload。容器服务会在集群的所有节点上使用同一名称创建数据卷。如下图所示。

数据卷列表						刷新创建
常见问题: 🔗 数据卷指南						
集群: test-link ▼						
0 节点	卷名	驱动	挂载点	引用容器	卷参数	操作
iZbp16vptdvm8y5jtu9l4jZ	f91423c7345bbc3cd7c09c78	本地磁盘	/var/lib/docker/volumes/	wordpress_web_1		删除所有同名卷
iZbp16vptdvm8y5jtu9l4jZ	fd23b180206446033b0e5d2c	本地磁盘	/var/lib/docker/volumes/	wordpress_web_1		删除所有同名卷
iZbp16vptdvm8y5jtu9l4jZ	wp_upload	OSS文件系统	/mnt/acs_mnt/ossfs/cjite	wordpress_web_1	查看	删除所有同名卷
iZbp135o869v7c5tmnc6dgZ	a03bbbe91cd847704654cc65	本地磁盘	/var/lib/docker/volumes/	wordpress_web_3		删除所有同名卷
iZbp135o869v7c5tmnc6dgZ	wp_upload	OSS文件系统	/mnt/acs_mnt/ossfs/cjite	wordpress_web_3	查看	删除所有同名卷
iZbp135o869v7c5tmnc6dgZ	775c1dd987160e6e512ad64c	本地磁盘	/var/lib/docker/volumes/	wordpress_web_3		删除所有同名卷
iZbp1148ey2z3dg94qdpa7Z	wp_upload	OSS文件系统	/mnt/acs_mnt/ossfs/cjlte	wordpress_web_2	查看	删除所有同名卷

2. 使用 OSSFS 数据卷。

WordPress 的附件,默认存放在 /var/www/html/wp-content/uploads 中。本示例中,只需将 OSSFS 数据卷映射到该目录,即可实现在不同的 WordPress 容器之间共享同一个 OSS bucket。

- i. 在 容器服务管理控制台, 在 Swarm 菜单下, 单击左侧导航栏中的应用。
- ii. 选择本示例中所使用的集群,选择本示例中所创建的应用wordpress 并单击右侧的变更配置。

容器服务		应用列表					刷新	创建应用
Swarm	Kubernetes		2					
概览		常见问题:	可创建应用		⑥ 简单路由蓝绿发布策略			
应用	1	集群: test	▼ 2 隐藏系统	应用			名称 ▼	
服务		应用名称	状态	服务数	创建时间 🔺	更新时间 ▲	3	操作
集群		wordpress	●运行中	2	2018-01-23 10:39:40	2018-01-23 10:42:11	变更配置 删除 重新部署	觱│ 事件

容器服务

iii. 在模板中添加 OSSFS 数据卷到 WordPress 目录的映射。

⑦ 说明 您必须修改应用版本,否则无法重新部署应用。

变更配置		>
应用名称: *应用版本:	wordpress 1.1 注意:提交配置变更需要您更新应用版本号,否则确定按钮无法点击	
应用描述:		
使用最新镜像:	■● 重新调度: ◎ ◎	
发布模式:	标准发布 🔻 🚳	
模板:	1 - web: 2 image: registry.aliyuncs.com/acs-sample/wordpress:4.3 3 - ports: 4 - '80' 5 volumes: 6 - 'wp_upload:/var/www/html/wp-content/uploads' 7 - environment: 8 WORDPRESS_AUTH_KEY: changeme 9 WORDPRESS_LOGGED_IN_KEY: changeme 10 WORDPRESS_LOGGED_IN_KEY: changeme 11 WORDPRESS_LOGGED_IN_KEY: changeme 12 WORDPRESS_LOGGED_IN_KEY: changeme 13 WORDPRESS_LOGGED_IN_SALT: changeme 14 WORDPRESS_SECURE_AUTH_SALT: changeme 15 WORDPRESS_NONCE_SALT: changeme 16 WORDPRESS_NONCE_AA: changeme 17 restart: always ✓ 使用已有编排模板 标签说明	
	确定	取消

iv. 单击确定, 重新部署应用。

3. 打开 WordPress, 上传附件, OSS bucket 里就能看到上传的附件了。
5.使用 Docker Compose 测试集群网络 连通性

本文档提供一个简单的 Compose file 来实现一键部署,您可以通过访问 endpoint 来检查容器间的互通性。

场景

在 Docker 集群中部署相互依赖的应用时,需要应用之间可以互相访问,实现跨 host 间容器网络的互通性。 而有时候因为网络问题,会导致 link 之间不能互相访问。类似情况下,很难定位发生错误的原因。因此,使 用一套简单易用的 Compose file 来测试当前集群网络各个主机上容器之间是否互通变得很重要。

解决方案

使用我们提供的镜像和编排模板测试容器间的互通性。

web: image: registry.aliyuncs.com/xianlu/test-link command: python test-link.py restart: always ports: - 5000 links: - redis labels: aliyun.scale: '3' aliyun.routing.port_5000: test-link; redis: image: redis restart: always

本示例使用 Flask 来完成测试功能。

以上编排模板部署了一个 Web 服务和一个 Redis 服务。Web 服务包含三个 Flask 容器。Flask 容器启动时, 会被平均分配到三个节点上运行。这就保证了三个容器在不同的宿主机上,只要能互相 ping 通,就说明当 前网络是可以实现容器跨主机互联的。Redis 运行在其中一台机器上,每个 Flask 容器启动后都会向 Redis 注 册,报告自己的 IP 地址。当三个 Flask 容器都启动完毕后, Redis 中也就有了集群中所有容器的 IP 地址。访 问其中任意一个 Flask 容器,该容器就会向另外两个容器发起 ping 命令,通过 ping 的结果确定当前集群的 网络连通性。

操作步骤

1. 创建一个包含三个节点的集群。

本示例中集群的名称为 test-link。有关如何创建集群的详细信息,参见创建集群。

说明 创建集群	时,请创建	建负载均衡实例	0							
集軒列表					您最多可以	(创建 5 个集群,每个集群。	最多可以添加 20 个节点	刷新	创建Swan	m集群
常见问题: ②如何创建集群 ③如何添	□巳有云服务器 🔗 跨可用	区节点管理 🔗 集成日志服务 🔗 i	通过Docker客户	靖连接集群						
名称 ▼										
集群名称/ID	集群类型 地域	网络类型	集群状态	节点状态 🕜	节点个数	创建时间	Docker版本			操作
test-link	阿里云集群 华东	虚拟专有网络 1	●运行中	健康 ℃	3	2018-01-22 13:11:34	17.06.2-ce	管理	查看日志 监控	删除 更多 •

2. 使用上面的模板,创建一个应用(本示例中创建名为test-cluster-link的应用),部署 web 服务和 redis 服务。

有关如何创建应用的详细信息,参见创建应用。

3. 在应用列表页面,单击应用的名称,查看创建的服务。

服务列表	容器列表 日志	事件 路由列表			
服务名称	所屬应用	服务状态	容器状态	镜像	擬作
redis	test	●运行中	运行中:1 停止:0	redis:latest	停止 重启 重新调度 变更配置 删除 事件
web	test	●运行中	运行中 :3 停止:0	registry.aliyuncs.com/xianlu/test-link:latest	停止 重启 重新调度 交更配置 删除 事件

4. 单击 web 服务的名称,进入服务详情页面。

可以看到,三个容器(test-cluster-link_web_1、test-cluster-link_web_2、test-cluster-link_web_3)都已经启动了,并且分布在不同的节点上。

基本信息											
服务名称: web		所在应用:	test (ii	像: registry.aliyuncs.com/xianlu/test-lir	容器数目: 3	●运行中					
访问端点: http://test	t-link.			n-hangzhou.alicontainer.com							
29番 日志 配置 事件											
名称/ID	状态	健康检测	镜像	端口	容器IP	节点IP		操作			
test_web_1 () 4130aa56f41cc164	running	正常	registry.aliyunc sha256:f5a856388		10.004	192.168.181.146	删除 停止 监控	日志 远程终端			
test_web_2 () 3f65175d058e4e4b	running	正常	registry.aliyunc sha256:f5a856388	-	1.0	192.168.181.147	删除 停止 监控	日志 远程终端			
test_web_3 () 59241239eb153807	running	正常	registry.aliyunc sha256:f5a856388	10,000,000,000,0070,000000	10.00	192.168.181.145	删除 停止 监控	日志 远程终端			

5. 访问 web 服务的访问端点。

从页面的反馈来看,容器 test-cluster-link_web_1 可以访问容器 test-cluster-link_web_2 和 test-cluster-link_web_3。

← → C ③ test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

current ip is 172.18.1.3 ping 172.18.1.3 response is True ping 172.18.2.4 response is True ping 172.18.3.3 response is True

刷新一下页面。可以看到,容器 test-cluster-link_web_2 可以访问容器 test-cluster-link_web_1 和 test-cluster-link_web_3。

- - - - C 🛭 🛈 test-link.c66d84378ce3a42dd8e22494da72f1563.cn-hangzhou.alicontainer.com

current ip is 172.18.2.4 ping 172.18.1.3 response is True ping 172.18.2.4 response is True ping 172.18.3.3 response is True

以上结果显示当前集群容器之间是互通的。

6.镜像服务

6.1. Docker 镜像的基本使用

Docker 的镜像存储中心通常被称为 Registry。

当您需要获取 Docker 镜像时,首先需要登录 Registry,然后拉取镜像。在您修改过镜像之后,您可以再次 将镜像推送到 Registry 中去。

基本概念

Docker的镜像地址是什么?我们来看一个完整的例子。以容器服务的公共镜像为例, registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8 。

- registry.cn-hangzhou.aliyuncs.com : Registry 域名。
- acs:命名空间。
- agent: 仓库名称。
- 0.8: Tag、镜像标签(非必须,默认为 latest)。

将这个几个完全独立的概念组合一下:

- registry.cn-hangzhou.aliyuncs.com/acs/agent : 仓库坐标。
- acs/agent : 仓库全名(通常在 API 中使用)。

基本使用

本文档的重点是介绍 Docker 最常用的三个命令: login、pull、push。

docker login

以阿里云杭州公网 Registry 为例。

登录时必须指明 Registry 域名,并输入您的用户名和登录密码。

⑦ 说明 此处的登录密码是您在 镜像仓库管理控制台 设置的,而不是您的阿里云登录密码。

Docker镜缭仓库	镜像仓库列表	亚太东北 1 (东京)	修改仓库密码		\times		修改docker登录	密码 创建镇像仓库
镜像列表	namespace筛选:	全部 *	docker登录时使用的F	1户名为阿里云账户全名,密码即为现在您设置的密码	-		1	
Namespace管理	仓库名称	namespace	*密码:			创建时间	9	操作
▼ 镜像库	test	test_wang	* 2000 1 327 22.		2	2017-03	3-20 15:20:47	管理 删除
镜像搜索			MB 0/ cd k2+				共有1条 ,每页显示:20条	« < 1 > »
我的收藏				统改成码	取消			
加速器				19-104-0418-3	100/10			

docker@default-online:~\$ docker login registry.cn-hangzhou.aliyuncs.com Username: sample@alibaba-inc.com Password: Login Succeeded

登录成功之后会显示 Login Succeeded 。

另外,您还可以通过查看 config.json 文件,确认您的登录信息。

docker pull

```
? 说明
```

- 如果您要拉取 Docker 官方的镜像,参考下方相关链接中的加速器文档。
- 如果您要拉取公共仓库下的镜像,不登录 Registry 也是可以拉取的。

以容器服务的公共镜像 registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8 为例。

docker@default-online:~\$ docker pull registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8 0.8: Pulling from acs/agent 5a026b6c4964: Already exists e4b621e8d9cb: Already exists 8bc2fd04bdd4: Pull complete a977b0087b3e: Pull complete 8f6e00ea13c6: Pull complete 8f6e00ea13c6: Pull complete 9c07bcabc35d: Pull complete 9c07bcabc35d: Pull complete Digest: sha256:cac848bd31bccf2a041bda7b57e3051341093abde6859df9ee9d332dfec6ddd9 Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8

您可以使用下边的命令查看下载下来的镜像(注意仓库坐标和 Tag)。

docker@default-online:~	-\$ docker ima	ages				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE		
registry.cn-hangzhou.ali	yuncs.com/a	cs/agent 0.8	b9ba584	1bdb0	24 hours ago	42.18 MB

docker push

镜像在本地环境构建或是打包好之后,就可以推到 Registry。

前提条件是,您有对这个仓库的读写权限或是读写授权。否则您会看到下面的报错信息。

docker@default-online:~\$ docker push registry.cn-hangzhou.aliyuncs.com/acs/agent:0.8 The push refers to a repository [registry.cn-hangzhou.aliyuncs.com/acs/agent] 359f80267111: Layer already exists 7e5fa28d90b8: Layer already exists b20d7f600f63: Layer already exists 4a159b4f8370: Layer already exists 7c3712ebe877: Layer already exists d91d130a53aa: Layer already exists fcad8ad5a40f: Layer already exists unauthorized: authentication required

FAQ

docker login 失败

主要需要排查以下两种可能。

 您使用了阿里云账户的登录密码,而不是 Registry 的独立登录密码。Registry 的登录密码是在 镜像仓库管 理控制台 上设置与修改的。

	「梅海へ度到去			修改仓库密码			×			Autor	
Docker镜像仓库	1 祝像包件列表	业太东北1(东京)	4	dockor發 思时体田的P	86々も原用二能66々、 肉田町も町左切込業の肉田				19 DOOCKer 登录图码	切建筑	876) 876)
镜像列表	namespace筛选:	全部	Ŧ	COCKET SEARCH SEARCH	2、白沙岭东水水、王母,在房间小水底下空的星站				1		
Namespace管理	仓库名称	namespace		*密码:				创建时间			招
▼ 镜像库	test	test_wang			1 2011 (2)XIBH 1-4. 10 3-404 (14)E2 (14)	2		2017-03-20 15:20:4	7	管理	1 89
镇像搜索				"确认密码:	******			共有1条,	每页显示:20条 « 、	1 >	
我的收藏			÷								
加速器					修改寄	〇日 取消					

 您使用了 sudo 进行登录。使用 sudo 时,系统第一个要求输入的密码是 Linux 的用户密码。您可能在这里 输入了 Registry 的登录密码,导致登录操作失败。

区分这个错误的方式很简单, Linux 的用户密码大多允许尝试三次,错误时会提示 try again。而 Registry 的登录密码错误一次之后就会退出,并返回以下错误。

Error response from daemon: Get https://registry.cn-hangzhou.aliyuncs.com/v2/: unauthorized: authenti cation required

docker pull 失败, 提示 Error: image xxx not found

主要的排查步骤。

- 如果您下载的是公共仓库,那么问题应该为镜像地址不正确。请在 镜像仓库管理控制台 搜索一下这个公共仓库,检查一下想要下载的这个镜像版本是不是真实存在。
- 您想要下载一个私有仓库中的镜像,这时首先确认一下 Registry 的登录状态。运行下边的命令,可以看到 所有登录的 Registry 域名。

cat ~/.docker/config.json

查看里面是不是包括您想要下载镜像的 Registry 域名。如果没有的话,您需要先进行登录操作。

如果显示已经登录的话,那么您需要确认您登录的这个账户是否有权限下载这个镜像。子账户默认没有任 何权限,参见下方相关链接中主子账户授权的文档。

docker push 失败

主要的排查步骤和 docker pull 基本一致,仅仅是授权要求的级别较 pull 更高一些。

6.2. 镜像服务主子账号授权

主子账号功能满足了企业账号分权管理的场景。子账号开通服务之后,主账号就可以对子账号进行授权。允 许指定的子账号拥有镜像的推送、拉取权限,或是修改仓库信息等功能。

子账号开通服务

在进行各类授权之前,您需要确认子账号已经开通服务。

⑦ 说明 在子账号开通服务时,默认是没有任何仓库或命名空间授权的。

1. 使用子账号登录镜像仓库管理控制台。

2. 填写 Docker 客户端的登录密码,即可完成服务的开通。

	傍海今府到主	初始化设置
Docker現像包库	106186131年7月-24	docker叠是附使用的用户全为同甲元账户全全, 家园即为现在你没有的家园
镜像列表	namespace筛选: 全部 \$	COOKERANDAUNU JIJI JIJI JIJA ANA EJI MANANDARANAN
Namespace管理		* 密码: 7-30位、必须包含字母、符号或数字中的至少面面
▼ 镜像库		
镜像搜索		, did // 45 (b);
我的收藏		
加速器		确定

为子账号提供仓库授权

子账户开通服务之后,可以用主账号登录 镜像仓库管理控制台 对子账号进行授权。

- 1. 使用主账号登录 镜像仓库管理控制台。
- 2. 选择一个镜像仓库,单击右侧的管理。

Docker镜像仓库	镜像仓库列表	亚太东北 1 (东京)	华东1 华东2	4 kdt1	华北2 华南1	欧洲中部 1 (法兰克福)	美国西部1(硅谷)		修改docker登录密码	创建镜像合库
镜像列表	namespace筛选:	全部	•							
Namespace管理	合库名称	namespace	性质	权限	合库地址			创建时间		操作
 镜像库 	test	test_	私有	管理	registry.cn-h	angzhou.aliyuncs.com/		2017-03-20 15:20:	47	管理 删除
加速器								共有1条,	每页显示:20条 « 、	$1 \rightarrow >$

3. 单击左侧导航栏中的仓库授权 > 添加授权。

<	test		
基本信息	仓库授权		
构建			
仓库授权	合库权限 namespace		
webhook	权限筛选: 全部 ▼	764	加授权
镜像版本	用户名	权限	操作
		① 没有查询到符合条件的记录	

4. 在弹出的对话框中,选择要授权的用户并为用户选择适当的权限,单击确定。

子账户需要登录控制台 建。	9开通服务之后7	「可以被授权,如需	≌创建新的子账户请登录R/	AM控制台创
搜索用户	搜索			
青选择用户		已选择		全部移除
昏无可选用户		yu	◎ 只读 ⑧ 读写	◎ 管理权限
	<			
	>			

您可以在仓库授权页面看到创建的授权信息,并可以对授权进行修改或删除。

使用子账号登录 镜像仓库管理控制台,您可以看到刚刚授权的镜像仓库。

Docker镜像仓库	镜像仓库列表	亚太东北 1 (东京)	华东1 华东	辰2 华北1	华北2 华南1 欧洲中部1(法兰	充福) 美国西部 1 (硅谷)		修改docker登录密码	创建镜像仓库
镜像列表	namespace筛选:	全部	•						
Namespace管理	合库名称	namespace	性质	权限	仓库地址		创建时间		操作
▶ 镜像库	test	test_	私有	读写	registry.cn-hangzhou.aliyuncs.com		2017-03-20 1	5:20:47	管理 删除
加速器							共有2	条 , 每页显示 : 20条 《	

为子账号提供命名空间级别的授权

如果需要将整个命名空间的权限都交给一个子账号,那么可以在命名空间管理中为子账号进行授权。

- 1. 使用主账号登录 镜像仓库管理控制台。
- 2. 单击左侧导航栏中的Namespace管理,选择一个命名空间并单击右侧的管理。

Docker镜像仓库	namespace實現		创建namespace
镜像列表	namespace	权限	1977 1986
Namespace管理	test	管理	管理
 镜像库 加速器 			共有5条,每页显示:20条 《 〈 1 〉 »

3. 单击添加授权。

<	test		添加授权
Namespacei羊情	权限筛选: 全部 ▼		
	用户名	权理	in the second s
		① 没有直询到符合条件的记录	

4. 在弹出的对话框中,选择要授权的用户并为用户选择适当的权限,单击确定。

涂加山化又PR			
子账户需要登录控制台开通 建。	服务之后才	可以被授权,如	口需创建新的子账户请登录RAM控制台创
搜索用户	搜索		
请选择用户		已选择	全部移除
暂无可选用户		yu	◎ 只读 ◎ 读写 ⑧ 管理权限
	< >		
			确定取消

您可以在命名空间授权页面看到创建的授权信息,并可以对授权进行修改或删除。 使用子账号登录 镜像仓库管理控制台,您可以看到刚刚授权的命名空间下所有的镜像仓库。 ⑦ 说明 如果您同时对子账号进行了命名空间和其下镜像仓库的授权,授权以更高级别的授权为准。本示例中,命名空间的授权为管理,镜像仓库的授权为读写,则取高级别的权限:管理。

子账号新建命名空间

子账号想要独立使用服务的话,也可以选择新建一个命名空间自己使用。

这个命名空间还是主账号所有的资源。新建的命名空间会自动为子账号授予管理权限。一个主账号的命名空间上限是五个。

- 1. 使用子账号登录 镜像仓库管理控制台。
- 2. 单击左侧导航栏中的Namespace管理并单击创建namespace。

Docker镜像仓库	namespace管理		创建namespace
镜像列表	namespace	权限	操作
Namespace管理	test_	管理	管理
 镜像库 			##58 每百見云 · 20 条 / / / / / / / /
加速器			

7.日志

7.1. 容器服务中使用 ELK

本文档介绍如何在容器服务里使用 ELK。

背景信息

日志是 IT 系统的重要组成部分,记录了系统在什么时候发生了什么事情。我们可以根据日志排查系统故障, 也可以做统计分析。

通常日志存放在本机的日志文件里,需要查看日志的时候,登录到机器上,用 grep 等工具过滤关键字。但 是当应用要部署在多台机器上的时候,这种方式查看日志就很不方便了,为了找到一个特定的错误对应的日 志,不得不登录到所有的机器上,逐个过滤文件。于是出现了集中式的日志存储方式:所有日志收集到日志 服务里,在日志服务里可以查看和搜索日志。

在 Docker 环境里,集中式日志存储更加重要。相比传统的运维模式,Docker 通常使用编排系统管理容器, 容器和宿主机之间的映射并不固定,容器也可能不断的在宿主机之间迁移,登录到机器上查看日志的方式完 全没法用了,集中式日志成了唯一的选择。

容器服务集成了阿里云日志服务,通过声明的方式自动收集容器日志到日志服务。但是有些用户可能更喜欢用 ELK(Elasticsearch+Logstash+Kibana)这个组合。本文档介绍如何在容器服务里使用 ELK。

整体结构



我们要部署一个独立的 logst ash 集群。logst ash 比较重,很耗资源,所以不会在每台机器上都运行 logst ash,更不要说每个 Docker 容器里。为了采集容器日志,我们会用到 syslog、logspout 和 filebeat, 当然您还可能会用到其他的采集方式。

为了尽可能贴合实际场景,这里我们创建两个集群:一个名为testelk的集群用来部署 ELK,一个名为 app 的集群用于部署应用。

操作步骤

⑦ 说明 本文档中创建的集群和负载均衡均需位于同一地域下。

步骤1创建负载均衡实例

为了能让其他服务向 logst ash 发送日志,我们需要在 logst ash 前面配置负载均衡。

- 1. 创建应用前, 登录 负载均衡管理控制台。
- 2. 创建一个公网类型的负载均衡实例。
- 3. 设置两条监听规则。其中一条设置前端和后端的端口映射 5000: 5000, 另一条设置端口映射 5044: 5044, 不用添加后端服务器。

配置!	监听		\times
2	1.基本配置	2.健康检查配置 > 3.配置成功	
	前端协议 [端 口] : *	TCP : 5000	
	后端协议 [端 口] : *	TCP : 5000	
	带宽峰值:	不限制 配置 使用流量计费方式的实例默认不限制带宽峰值;峰值输入范围1- 5000	
	调度算法:	加权於询	
	使用服务器组: ❷	0	
	 展开高级配 — 置 		
		下一步取河	۴

步骤 2 部署 ELK

1. 登录 容器服务管理控制台, 创建集群 testelk。

有关如何创建集群,参见创建集群。

⑦ 说明 集群必须和上边创建的负载均衡实例位于同一地域。

2. 为集群绑定上边所创建的负载均衡实例。

在集群列表页面,选择集群 testelk,单击右侧的管理,单击左侧导航栏中的负载均衡 > 单击绑定SLB > 选择上边创建的负载均衡实例并单击确定。

3. 使用下面的编排模板部署 ELK。本示例创建了一个名为 elk 的应用。

有关如何使用编排模板创建应用,参见创建应用。

⑦ 说明 您需要使用您上边所创建的负载均衡实例的 ID 替换编排文件中的 \${SLB_ID}。

version: '2' services: elasticsearch: image: elasticsearch kibana: image: kibana environment: ELASTICSEARCH_URL: http://elasticsearch:9200/ labels: aliyun.routing.port_5601: kibana links: - elasticsearch logstash: image: registry.cn-hangzhou.aliyuncs.com/acs-sample/logstash hostname: logstash ports: - 5044:5044 - 5000:5000 labels: aliyun.lb.port_5044: 'tcp://\${SLB_ID}:5044' #先创建slb aliyun.lb.port_5000: 'tcp://\${SLB_ID}:5000' links: - elasticsearch

在这个编排文件里, Elast icsearch 和 Kibana 我们直接使用了官方镜像,没有做任何更改。logst ash 需 要配置文件,需要自己做一个镜像,把配置文件放进去。镜像源码参见demo-logst ash。

logstash 的配置文件如下。这是一个非常简单的 logstash 配置,我们提供了 syslog 和 filebeats 两种 输入格式,对外的端口分别是 5044 和 5000。

```
input {
  beats {
    port => 5044
   type => beats
  }
  tcp {
    port => 5000
   type => syslog
 }
}
filter {
}
output {
  elasticsearch {
    hosts => ["elasticsearch:9200"]
  }
  stdout { codec => rubydebug }
}
```

4. 配置 kibana index。

i. 访问 kibana。

URL 可以在应用的路由列表(单击所创建的应用的名称 elk,单击路由列表页签 > 单击路由地址) 里找到。

服务列表	容器列表	日志	事件	路由列表	
路由地址(集群绑定SLB后路由地址才能被访问) 操作					
kibana					
47.96. 设置服务权重					

ii. 创建 index。

根据您的实际需求进行配置并单击Create。

kibana	Management / Kibana					
KIDalla	Index Patterns Saved Objects Advance	Index Patterns Saved Objects Advanced Settings				
Ø Discover	Warning No default index pattern. You must					
U Visualize	select or create one to continue.	Configure an index pattern				
Oashboard						
3 Timelion		In order to use numary our must compare at least one more patterns met patterns are used to nemary the classicisement more to the num analytics against, mey are about so compare news.				
🔎 Dev Tools						
. Management		Use event times to create index names (DEPRECATED)				
		Index name or pattern				
		Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*				
		logstash-*				
		Do not expand index pattern when searching (Not recommended)				
		By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.				
		Searching against the index pattern logstash-* will actually query elasticsearch for the specific matching indices (e.g. logstash-2015.12.21) that fall within the current time range.				
		Unable to fetch mapping. Do you have indices matching the pattern?				

步骤 3 收集日志

Docker 里标准的日志方式是用 Stdout,所以我们先演示如何把 Stdout 收集到 ELK。如果您在使用文件日志,可以直接用 filebeat。我们用 wordpress 作为演示用的例子,下面是 wordpress 的编排模板。我们在 另外一个集群中创建应用 wordpress。

1. 登录 容器服务管理控制台, 创建集群 app。

有关如何创建集群,参见创建集群。

- ⑦ 说明 集群必须和上边创建的负载均衡实例位于同一地域。
- 2. 使用下面的编排模板创建应用wordpress。

⑦ 说明 您需要使用您上边所创建的负载均衡实例的 IP 替换编排文件中的 \${SLB_IP}。

version: '2' services: mysql: image: mysql environment: - MYSQL_ROOT_PASSWORD=password wordpress: image: wordpress labels: aliyun.routing.port_80: wordpress links: mysql:mysql environment: - WORDPRESS_DB_PASSWORD=password logging: driver: syslog options: syslog-address: 'tcp://\${SLB_IP}:5000'

待部署成功后,找到 wordpress 的访问地址(单击所创建的 wordpress 应用的名称 > 单击路由列 表页签 > 单击路由地址),即可访问 wordpress 应用。

3. 在应用列表页面,单击应用elk>单击路由列表页签>单击路由地址。

成功访问 kibana 的页面,可以查看已经收集到的日志。

21 hits			New Save Open Share < 🔿 Last 15 minutes 🗲
Search (e.g. status:200 AND extensio	n:PHP)		Uses lucene query syntax
Add a filter 🕈			
logstash-*	0	November 13th 2017, 17:04:32.157 - November 13th 2017, 17:19:32.157 Auto	Ŧ
Selected Fields ?source	15 10		0
Available Fields	5-		
② @timestamp	0 170500 170500 170	100 170800 170900 171000 171100 171200 171800 171400	17:1500 17:1600 17:17:00 17:18:00 17:19:00
t @version	0	@timestamp per 30 seconds	
t _id	Time 🗸	source	
t _index	 November 13th 2017, 17:18:43.813 	<pre>nexxage: <30>Nov 13 17:18:43 e98d6bb157be[15274]: 172.18.5.2 = [13/Nov/2017:09:18:43 +00</pre>	00] "GET /favicon.ico HTTP/1.1" 200 192 "http://wordpress.ccOce
# _score		fac5cd444816b8fe6949af555f91.cn-hangzhou.alicontainer.com/wp-admin/install.php?step=1" "No.	zilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTM
t _type		L, like Gecko) Chrome/61.0.3163.100 Safari/537.36" Bversion: 1 Btimestamp: November 13th 20	17, 17:18:43.813 kext: 47.96.167.189 pert: 42,096 type: syslog
t host		_id: AV-OrHzgn446ypiV1c _type: syslog _imdex: logstash-2017.11.13 _seere: -	
t message	 November 13th 2017, 17:18:43.713 	<pre>nessage: <30>Nov 13 17:18:43 e98d6bb157be[15274]: 172.18.5.2 - [13/Nov/2017:09:18:43 +00</pre>	00] "POST /wp-admin/install.php?step=1 HTTP/1.1" 200 2867 "htt
# port		p://wordpress.cc0cefac5cd444816b8fe6949af555f91.cn-hangzhou.alicontainer.com/wp-admin/inst	all.php" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/
t type		537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36" @version: 1 @timestamp: Nove	mber 13th 2017, 17:18:43.713 hest: 47.96.167.189 pert: 42,096

7.2. Docker 日志收集新方案: log-pilot

本文档介绍一款新的 Docker 日志收集工具: log-pilot 。log-pilot 是我们为您提供的日志收集镜像。您可以 在每台机器上部署一个 log-pilot 实例,就可以收集机器上所有 Docker 应用日志。(注意: 只支持Linux版本 的Docker,不支持Windows/Mac版)。

log-pilot 具有如下特性:

- 一个单独的 log 进程收集机器上所有容器的日志。不需要为每个容器启动一个 log 进程。
- 支持文件日志和 stdout。docker log dirver 亦或 logspout 只能处理 stdout, log-pilot 不仅支持收集 stdout 日志,还可以收集文件日志。
- 声明式配置。当您的容器有日志要收集,只要通过 label 声明要收集的日志文件的路径,无需改动其他任何配置, log-pilot 就会自动收集新容器的日志。
- 支持多种日志存储方式。无论是强大的阿里云日志服务,还是比较流行的 elasticsearch 组合,甚至是 graylog, log-pilot 都能把日志投递到正确的地点。

 开源。log-pilot 完全开源,您可以从 Git项目地址 下载代码。如果现有的功能不能满足您的需要,欢迎提 issue。

快速启动

下面演示一个最简单的场景:先启动一个 log-pilot,再启动一个 tomcat 容器,让 log-pilot 收集 tomcat 的日志。为了简单起见,这里先不涉及阿里云日志服务或者 ELK。如果您想在本地运行,只需要有一台运行 Docker 的机器就可以了。

首先启动 log-pilot。

⑦ 说明 以这种方式启动,由于没有配置后端使用的日志存储,所有收集到的日志都会直接输出到控制台,所以主要用于调试。

打开终端, 输入命令:

docker run --rm -it \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v /:/host \
 --privileged \
 registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat

您会看到 log-pilot 的启动日志。

root@c 3-nodel:/# docker runrm -it \
> -v /var/run/docker.sock:/var/run/docker.sock \
> -v /:/host \
>privileged \
> registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
Unable to find image 'registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat' locally
0.9.5-filebeat: Pulling from acs-sample/log-pilot
a073c86ecf9e: Pull complete
7ba3e804adbd: Pull complete
7bff0b2064d3: Pull complete
ee47809ba289: Pull complete
070dlb641126: Pull complete
Digest: sha256:427b5d81168a5f6584f063a814709618d7b81ed34f961dcd58d223314602b987
Status: Downloaded newer image for registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
enable pilot: filebeat
use default output
DEBU[0000] 72c3eb36e84c2a52f4b309b6c700401e62f8357ba757c06406aa6f3d4aabc519 has not log config, skip
DEBU[0000] b13742566befe366d6676e2d96f8f5641d5947aaf0e481683613d6cedb268066 has not log config, skip
DEBU[0000] fdbaad5815ea4630e56ffb6e3e02e04a0a33a8a2fb26c7635c470b10c440d454 has not log config, skip
DEBU[0000] 84b359b1f8800330748903eb7e091c1020b9732aa714e70a971ad7c9beb1eb15 has not log config, skip
DEBU[0000] 729e8ecad43f02105142bfbd447613766b3661236554b7048b58be13f0c57b6a has not log config, skip
DEBU[0000] b9fac7428bb6e75bd6ccbdabe5a76f5e7d348e5a004cb160e9063d52da34c927 has not log config, skip
DEBU[0000] ce02ee3db38462779f55a2d05c413365465d636e271a9c0881acc32e36981d16 has not log config, skip
DEBU[0000] 857e94936233dc02c5dae6c866ddfb0f3a2877a487079c0b7c2085bfb43fc947 has not log config, skip
INFO[0000] starting filebeat
INFO[0000] filebeat started: 33
INFO[0000] Reload gorouting is ready
INFO[0000] filebeat watcher start

不要关闭终端。新开一个终端启动 tomcat。tomcat 镜像属于少数同时使用了 stdout 和文件日志的 Docker 镜像,非常适合这里的演示。

```
docker run -it --rm -p 10080:8080 \
-v /usr/local/tomcat/logs \
--label aliyun.logs.catalina=stdout \
--label aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt \
tomcat
```

说明:

```
• aliyun.logs.catalina=stdout 告诉 log-pilot 这个容器要收集 stdout 日志。
```

aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt 则表示要收集容器内 /usr/local/tomcat/logs/目录下所有名字匹配 localhost_access_log.*.txt 的文件日志。后面会详细介绍 label 的用法。

 ⑦ 说明 如果您在本地部署 tomcat,而不是在阿里云容器服务上,您需要指定 -v /usr/local/tomcat/logs; 否则 log-pilot 没法读取到日志文件。容器服务已经自动做了优化,不需自己加 -v 了。

log-pilot 会监控 Docker 容器事件,当发现带有 aliyun.logs.xxx 容器时,自动解析容器配置,并且开始收集对应的日志。启动 tomcat 之后,您会发现 log-pilot 的终端立即输出了一大堆的内容,其中包含 tomcat 启动时输出的 stdout 日志,也包括 log-pilot 自己输出的一些调试信息。

DEBU[1043] Process container start event: 2607cba67709075e373009751664734caa7b665397660087431240608540 1100[1043] [Doc: 2607cba67750720975709759739075975976008743124005304 = 4[accs: Abst/schervolumes/a6dd39848a81263028ae71408546c27641c7731793060857a6a8f18fc14b978/_data
/usr/tocat/togs nonex map[time_key] timestamp] Locatnost access Log.* Txt map[noex:access top1:access] true false}
INPUT4451 [dg]: $209/cba0br/94c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e00095540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e00095540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e00095540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e00095540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e0009540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/809/c510b4/54caa/ebbc/39/ebb0874151e00095540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/ebb/39/ebb0874151e0009540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/39/ebb08740009540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/ebb/39/ebb0874051e0009540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/ebb/39/ebb0874051e0009540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/194c/3e8/3/ebb/39/ebb08740000540 = 61catatina /nost/var/(10)/d0cker/containers/209/cba0br/39/ebb/$
nonex map(timetionmation-on-on-on-on-on-on-on-on-on-on-on-on-o
In 6(1465) Reduct release
DEBUILING Lot pred to related filebest
Babo (1405) Not need to head to head the start of the sta
"catalina" "docker container": "tender innes". "index": "catalina" "offset :113. "stream : "stdout". "source": "/host/var/lb/docker/containers/2b87c5b86725987259897c5b669749675e87378907c5b66974975e8737897c5b66974975e8775697697697697697697697697697697697697
04605540/2b02cbab67209c75827378092c51664754ca32eb6c392e6608f4151e0d605540-ison.log", beat "{"rane": "72c3eb36684c", "best name": "72c3eb36684c", "best name"; "72c3eb36684c", "best name"; "72c3eb36684c", "best name"; "72c3eb36684c", "best name; "72c3eb36684c", "best name; "72c3eb36684c", "best name; "72c3e
{"@timestamp":"2018-10-09T03:58:07.0762"."@metadata":{"beat":"filebeat"."type":"doc"."yersion":"6.1.1"}."offset":225."stream":"stdout"."prospector":{"type":"log"}."topic":"catalina"."docker con
tainer": "tender jones", "index": "catalina", "beat": {"hostname": "72C3eb36e84c", "version": "6.1.1", "name": "72C3eb36e84c"}, "source": "/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754
caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540-json.log", "message": "Using CATALINA HOME: /usr/local/tomcat")
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"prospector":{"type":"log"},"index":"catalina","beat":{"name":"72c3eb36e84c","hostname":"
72c3eb36e84c", "version": "6.1.1"}, "source": "/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6608f41
51e0d605540-json.log","stream":"stdout","message":"Using CATALINA_TMPDIR: /usr/local/tomcat/temp","topic":"catalina","docker_container":"tender_jones","offset":343}
{"@timestamp":"2018-10-09T03:58:07.0762", "@metadata":{ beat": filebeat", "type":"doc", "version": "6.1.1"}, "source":"/host/var/lib/docker/containers/2b07cbab6f709c75e87378097c51664754caa7eb6c397e6
6081411s0e0605540;2b07cbab67706c7580370807c51664734caa7eb6c337e660814151e0d605540;2b01g, beat:{frame:rrZc2eb36e84c", hostname:rrZc3eb36e84c", hversunr:e6.ll*), message*:Usung JRE_HOME / docker-java-homepire", orfiset:i460, stream: "stdout", prospector: {frype:rlog", trcatlana", "docker_container":tender", ones ", index: "rcatlana"}
{"@timestamp":"2018-10-09T03:58:07.076Z","@metadata":{"beat":"filebeat","type":"doc","version":"6.1.1"},"docker_container":"tender_jones","index":"catalina","beat":{"name":"72c3eb36e84c","hostn
ame": 72c3eb36e84c", version": "6.1.1"}, "stream": "stdout", "offset":629, "prospector": {"type": "log"}, "message": "Using CLASSPATH: //usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tom
cat-juli.jar", "source": "/host/var/lib/docker/containers/2b07cbab6†709c75e87378097c51664754caa7eb6c397e6608†4151e0d605540/2b07cbab6†709c75e87378097c51664754caa7eb6c397e6608†4151e0d605540-json.lo
g","topic":"catalina"}
{"dtlmestamp": 2018-10-09103:58:07.552", "dmetadata": {"beat": "flebeat", "type": "doc", "version": "6.1.1"}, "offset": 836, "stream": "stdout", "message": 09-001-2018 03:58:07.553 IN+0 [main] org.apache
.catalina.startup.versionLoggerListener.log Server version: Apache Tomcat/8.5.34*, prospector::type::togr/, topic::catalina, index::catalina, "Deat it nostname": /2:3e030e84c , ver
\$10n; "b.l.1; "name: "/2C36036884("), \$50rce": "/N05T/Var/(LD/d0cKer/Container5/200/CDabbt/09C/588/3/809/C51604/54Caa/ebbc39/ebb08t4151e00605540/2D0/CDabbt/09C/588/3/809/C51604/54Caa/ebbc39/ebb0
araineodoosado joon.tog , ootke _container : tender_jones / Jatimastantino 0.000000000 (Stender Stender Jahan) (Maaraine Utani (Stender Stender
(withestamp - 2010-10-00-10-10-00-10-00-2) (we cauca - 10-00-00-00-00-00-00-00-00-00-00-00-00-0
ange o volocio volo volo volo volo volo volo volo vo
Beg 75e 87378097c51664754caa7ebfc392e6f608f4151e0f605540-ison.loc"}
{"@timestamp":"2018-10-09T03:58:07.6562"."@metadata":{"beat":filebeat"."type":"doc"."yersion":"6.1.1"}."stream":"stdout"."beat":{"name":"72c3eb36e84c"."hostname":"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat:"filebeat
1*], "index": "catalina", "source": "/host/var/lib/docker/containers/2007cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d605540/2007cbab6f709c75e87378097c51664754caa7eb6c397e6608f4151e0d60554
0-json.log", "offset":1241, "message": "09-Oct-2018 03:58:07.656 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number: 8.5.34.0", "prospector": {"type": "log"}, "top"
ic":"catalina"."docker container":"tender jones"}

您可以打开浏览器访问刚刚部署的 tomcat, 您会发现每次刷新浏览器, 在 log-pilot 的终端里都能看到类似的记录。其中 message 后面的内容就是从 /usr/local/tomcat/logs/localhost_access_log.XXX.txt 里收集到的日志。

使用 ElasticSearch + kibana

首先要部署一套 ElastichSearch + kibana,您可以参考 容器服务中使用 ELK 在阿里云容器服务里部署 ELK,或者按照 ElasticSearch/kibana 的文档直接在机器上部署。本文档假设您已经部署好了这两个组件。

如果您还在运行刚才启动的 log-pilot,先关掉,然后使用下面的命令启动。

```
⑦ 说明 执行之前,先把 ELASTICSEARCH_HOST 和 ELASTICSEARCH_PORT 两个变量替换成您实 际使用的值。 ELASTICSEARCH_PORT 一般为 9200。
```

```
docker run --rm -it \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v /:/host \
    --privileged \
    -e FLUENTD_OUTPUT=elasticsearch \
    -e ELASTICSEARCH_HOST=${ELASTICSEARCH_HOST} \
    -e ELASTICSEARCH_PORT=${ELASTICSEARCH_PORT} \
    registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
```

相比前面启动 log-pilot 的方式,这里增加了三个环境变量:

- FLUENTD_OUTPUT=elasticsearch : 把日志发送到 ElasticSearch。
- ELASTICSEARCH_HOST=\${ELASTICSEARCH_HOST} : ElasticSearch 的域名。
- ELASTICSEARCH_PORT=\${ELASTICSEARCH_PORT}: ElasticSearch的端口号。

继续运行前面的 tomcat,再次访问,让 tomcat 产生一些日志,所有这些新产生的日志都将发送到 ElasticSearch 里。

打开 kibana,此时您还看不到新日志,需要先创建 index。log-pilot 会把日志写到 Elast icSearch 特定的 index下,规则如下:

如果应用上使用了标签 aliyun.logs.tags , 并且 tags 里包含 target , 使用 target 作为 ElasticSearch 里的 index。否则,使用标签 aliyun.logs.XXX 里的 XXX 作为 index。

在前面 tomcat 里的例子里,没有使用 aliyun.logs.tags 标签,所以默认使用了 access 和 catalina 作为 index。我们先创建 index access 。

	kibana	Management / Kibana		
	KIDalla	Index Patterns Saved	Objects Advanced Settings	
Ø		Warning No default index		
ы		pattern. You must select or create one to continue.	Configure an index pattern	
\odot			In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and	
8			analytics against. They are also used to configure fields.	
بر				
۵	Management		Index contains time-based events Use event times to create index names (DEPRECATED)	
			Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash.*	
			access	
			Time-field name 🚯 refresh fields	
			@timestamp t	2
			Create	
			云海社区 yqualiyun.co	

创建好 index 就可以查看日志了。

	kibana	2 hits		New Save	Open Share	O February 9th 2017, 19:47:00.	000 to February 9th 2017, 19:47:30.000
	KIDalla	*					٩
Ø	Discover	access	0	February 9th 2017, 19:4	7:00.000 - February 91	:h 2017, 19:47:30.000 — <u>by second</u>	
101		Selected Fields	2 -				0
\odot		? _source	1.5 - 2				
8		Available Fields	-1 O				
۶		 ⊘ @timestamp <i>t</i>_id 	0.5 -				
•		t_index	19:47:05	19:47:10	19:47:15 @timestam	19:47:20	19:47:25
		# _score	C Time	5011750	editostani	p per accond	
		ℓ_type	Time +	_source			
		t docker_container	 February 9th 2017, 19:47:16.8 	69 message: 192.168.	2.1 [09/Feb/2	017:11:47:08 +0000] "GET / HT	TP/1.1" 200 11250 @timestamp: Fe
		£ host		bruary 9th 2017, 1	19:47:16.869 host	: jjz docker_container: log-	test _id: AVoisvVWXnslZ5_GvUrj
		t message		_type: fluentd _	index: access _s	core: -	
			 February 9th 2017, 19:47:16.8 	69 message: 192.168	2.1 [09/Feb/2	017:11:47:09 +0000] "GET / HT	TP/1.1" 200 11250 @timestamp: Fe
				bruary 9th 2017, 1	19:47:16.869 host	: jjz docker_container: log-	test _id: AVoisvVWXnslZ5_GvUrk
				_type: fluentd _	index: access _s	core: -	
							vg.aliyun.com

在阿里云容器服务里使用 log-pilot

容器服务专门为 log-pilot 做了优化,最适合 log-pilot 运行。

要在容器服务里运行 log-pilot , 您仅需要使用下面的编排文件创建一个新应用。有关如何创建应用,参见 创建应用。 pilot:

```
image: registry.cn-hangzhou.aliyuncs.com/acs-sample/log-pilot:0.9.5-filebeat
volumes:
        - /var/run/docker.sock:/var/run/docker.sock
        - /:/host
privileged: true
environment:
FLUENTD_OUTPUT: elasticsearch #按照您的需要替换
ELASTICSEARCH_HOST: ${elasticsearch} #按照您的需要替换
ELASTICSEARCH_PORT: 9200
labels:
        aliyun.global: true
```

接下来,您就可以在要收集日志的应用上使用 aliyun.logs.xxx 标签了。

label 说明

启动 tomcat 时,声明了下面两个 label 来告诉 log-pilot 这个容器的日志位置。

--label aliyun.logs.catalina=stdout

--label aliyun.logs.access=/usr/local/tomcat/logs/localhost_access_log.*.txt

您还可以在应用容器上添加更多的标签。

- aliyun.logs.\$name = \$path
 - 变量 name 是日志名称,只能包含 0~9、a~z、A~Z 和连字符(-)。
 - 变量 oath 是要收集的日志路径,必须具体到文件,不能只写目录。文件名部分可以使用通配符,例如, /var/log/he.log 和 /var/log/*.log 都是正确的值,但 /var/log 不行,不能只写到目录。 stdout 是一个特殊值,表示标准输出。
- aliyun.logs.\$name.format : 日志格式,目前支持以下格式。
 - none: 无格式纯文本。
 - json: json 格式,每行一个完整的 json 字符串。
 - csv: csv 格式。
- aliyun.logs.\$name.tags:上报日志时,额外增加的字段,格式为 k1=v1,k2=v2,每个 key-value 之间使 用逗号分隔,例如 aliyun.logs.access.tags="name=hello,stage=test",上报到存储的日志里就会出现 na me 字段和 stage 字段。

如果使用 Elast icSearch 作为日志存储, target 这个 tag 具有特殊含义,表示 Elast icSearch 里对应的 index。

扩展 log-pilot

对于大部分用户来说, log-pilot 现有功能足以满足需求, 如果遇到没法满足的场景, 您可以:

- 到 https://github.com/AliyunContainerService/log-pilot 提交 issue。
- 直接改代码, 再提 PR。

8.Docker 容器健康检查机制

在分布式系统中,经常需要利用健康检查机制来检查服务的可用性,防止其他服务调用时出现异常。自1.12 版本之后,Docker引入了原生的健康检查实现。本文将介绍Docker容器健康检查机制。

对于容器而言,最简单的健康检查是进程级的健康检查,即检验进程是否存活。Docker Daemon 会自动监控 容器中的 PID1 进程,如果 docker run 命令中指明了 rest art policy,可以根据策略自动重启已结束的容 器。在很多实际场景下,仅使用进程级健康检查机制还远远不够。比如,容器进程虽然依旧运行却由于应用 死锁无法继续响应用户请求,这样的问题是无法通过进程监控发现的。

Kubernetes 提供了 Liveness 与 Readness 探针分别对 Container 及其服务健康状态进行检查。阿里云容器服务也提供了类似的 服务健康检查机制。

Docker 原生健康检查能力

自 1.12 版本之后, Docker 引入了原生的健康检查实现,可以在 Dockerfile 中声明应用自身的健康检测配置。 HEALTHCHECK 指令声明了健康检测命令,用这个命令来判断容器主进程的服务状态是否正常,从而 比较真实的反应容器实际状态。

HEALTHCHECK 指令格式:

- HEALTHCHECK [选项] CMD <命令> : 设置检查容器健康状况的命令。
- HEALTHCHECK NONE : 如果基础镜像有健康检查指令,使用这行可以屏蔽。

⑦ 说明 在 Dockerfile 中 HEALTHCHECK 只可以出现一次,如果写了多个,只有最后一个生效。

使用包含 HEALTHCHECK 指令的 dockerfile 构建出来的镜像,在实例化 Docker 容器的时候,就具备了健康 状态检查的功能。启动容器后会自动进行健康检查。

HEALTHCHECK 支持下列选项:

- --interval=<间隔>:两次健康检查的间隔,默认为 30 秒。
- --timeout=<间隔>:健康检查命令运行超时时间,如果超过这个时间,本次健康检查就被视为失败,默认 30 秒。
- --retries=<次数>: 当连续失败指定次数后,则将容器状态视为 unhealt hy, 默认 3 次。
- --start-period=<间隔>:应用的启动的初始化时间,在启动过程中的健康检查失效不会计入,默认 0 秒 (从 V17.05 引入)。

在 HEALTHCHECK [选项] CMD 后面的命令,格式和 ENTRYPOINT 一样,分为 shell 和 exec 格式。命令的 返回值决定了该次健康检查的成功与否:

- 0: 成功
- 1: 失败
- 2:保留值,不要使用

容器启动之后,初始状态会为 starting (启动中)。Docker Engine 会等待 interval 时间,开始执行健康 检查命令,并周期性执行。如果单次检查返回值非 0 或者运行需要比指定 timeout 时间还长,则本次检查 被认为失败。如果健康检查连续失败超过了 retries 重试次数,状态就会变为 unhealthy (不健康)。

- 一旦有一次健康检查成功, Docker 会将容器置回 healthy (健康)状态。
- 当容器的健康状态发生变化时, Docker Engine 会发出一个 health_status 事件。

假设我们有个镜像是个最简单的 Web 服务,我们希望增加健康检查来判断其 Web 服务是否在正常工作,我 们可以用 curl 来帮助判断,其 Dockerfile 的 HEALTHCHECK 可以这么写: FROM elasticsearch:5.5 HEALTHCHECK --interval=5s --timeout=2s --retries=12 \ CMD curl --silent --fail localhost:9200/_cluster/health || exit 1

docker build -t test/elasticsearch:5.5 . docker run --rm -d \ --name=elasticsearch \ test/elasticsearch:5.5

我们可以通过 docker ps ,发现过了几秒之后, Elast icsearch 容器从 starting 状态进入了 healt hy 状态。

\$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES c9a6e68d4a7f test/elasticsearch:5.5 "/docker-entrypoin..." 2 seconds ago Up 2 seconds (health: starti ng) 9200/tcp, 9300/tcp elasticsearch \$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES c9a6e68d4a7f test/elasticsearch:5.5 "/docker-entrypoin..." 14 seconds ago Up 13 seconds (healthy) 9 200/tcp, 9300/tcp elasticsearch

另外一种方法是在 docker run 命令中,直接指明 healt hcheck 相关策略。

```
$ docker run --rm -d \
--name=elasticsearch \
--health-cmd="curl --silent --fail localhost:9200/_cluster/health || exit 1" \
--health-interval=5s \
--health-retries=12 \
--health-timeout=2s \
elasticsearch:5.5
```

为了帮助排障,健康检查命令的输出(包括 stdout 以及 stderr)都会被存储于健康状态里,可以用 docker inspect 来查看。我们可以通过如下命令,来获取过去5个容器的健康检查结果。

docker inspect --format='{{json .State.Health}}' elasticsearch

或

docker inspect elasticsearch | jq ".[].State.Health"

示例结果如下:

```
{
    "Status": "healthy",
    "FailingStreak": 0,
    "Log": [
    {
        "Start": "2017-08-19T09:12:53.393598805Z",
        "End": "2017-08-19T09:12:53.452931792Z",
        "ExitCode": 0,
        "Output": "..."
    },
    ...
}
```

由于应用的开发者会更加了解应用的 SLA,一般建议在 Dockerfile 中声明相应的健康检查策略,这样可以方 便镜像的使用。对于应用的部署和运维人员,可以通过命令行参数和 REST API 针对部署场景对健康检查策 略按需进行调整。

Docker社区提供了一些包含健康检查的实例镜像,我们可以在如下项目中获取 https://github.com/docker-library/healthcheck。

- ? 说明
 - 阿里云容器服务同时支持 Docker 原生健康检测机制和阿里云的扩展检查机制。
 - 目前 Kubernetes 还不提供对 Docker 原生健康检查机制的支持。

9.一键部署 Docker Datacenter

DDC 简介

Docker Datacenter (DDC) 是 Docker 发布的企业级容器管理和服务部署的整体解决方案平台。DDC 由以下 三个组件构成:

- Docker Universal Control Plane (Docker UCP): 一套图形化管理界面。
- Docker Trusted Registry (DTR): 授信的 Docker 镜像仓库。
- Docker Engine 商业版:提供技术支持的 Docker 引擎。

DDC 在 Docker 官网的地址为。

<mark>⊕ docker</mark> ⊗ CERTIFIED Plugin		ای ک م			
CON		CLE WORKFLOW			
PRIVATE IMAGE REGISTRY	SECURE ACCESS A MANAGEME	ND USER A	APPLICATION AND CLUSTER MANAGMENT	INTEGRATED LIFECYCLE	
IMAGE SECURITY SCANNING AND CONTINUOUS MONITORING	CONTENT TRUS VERFICATIO	T AND	POLICY MANAGMENT	MANAGEMENT	
SECURITY	NETWORK	0	VOLUMES	CONTAINER	
DISTRIBUTED STATE	CONTAINER RUI	NTIME	ORCHESTRATION	ENGINE	
Certified Infrastructure		OPERATING SYSTEMS	CLOUD	CERTIFIED	

DDC 与 Docker 公司的另外一个在线产品 Docker Cloud 对应。不过 DDC 主要针对企业用户在内部部署。用 户注册自己的 Dokcer 镜像到 DTR, UCP 管理整个 Docker 集群。并且这两个组件都提供了 Web 界面。

使用 DDC 需要购买 Licence,但是 Docker 公司提供了一个月的试用 Licence,可以在 Docker 官网注册后直 接下载。

DDC 部署架构



在上面的基础架构图里, Controller 主要运行 UCP 组件, DTR 运行 DTR 组件, Worker 主要运行客户自己的 Docker 服务。整个 DDC 环境都部署在 VPC 网络之下,所有的 ECS 加入同一个安全组。每个组件都提供了一 个负载均衡,供外网访问。而运维操作则是通过跳板机实现。另一方面为了提升可用性,整个 DDC 环境都是 高可用部署,也就是说 Controller 至少有两台,同理 DTR 也至少有两台。

DDC 一键部署

您可以使用阿里云资源编排 ROS 通过下面的链接一键部署 DDC。

一键部署 DDC

上边的编排模板默认会在华北 2 地域部署 DDC。如果您需要调整地域,请单击页面右下角的上一步,然后重新选择地域,然后单击下一步。

填写如下图中必填的信息或者根据您的需求调整信息后,单击创建就可以部署一套 DDC。

A VEH SELES	启动代	19建成40
已选地域:	44개 2	
* 栈空 🕜 :	alivun docker datacenter v1	7
	长度1-64个字符,以龙小写字母开头,可包含数字,"_"或"-" 栈名不能重复,创建后不能修改	
* 创建超时 (分钟) 🖉 :	60	
	以分钟为甲位的止整数,数字沧围 10-180	
	✓ 失败回滚	
DTRInstanceType :	ecs.n4.large	T
ControllerSlaveMaxAmount @ :	0	r
ControllerSystemDiskCategory :	cloud_ssd	r
ControllerInstanceType :	ecs.n4.large	T
WorkerSystemDiskCategory :	cloud_ssd	T
DTRSystemDiskCategory :	cloud_ssd	r
WorkerMaxAmount :	1	
ControllerImageId :	ubuntu_14_0405_64_40G_alibase_20170525.vhd	
WorkerImageId :	ubuntu_14_0405_64_40G_alibase_20170525.vhd	
WorkerInstanceType :	ecs.n4.large	r
* UCPAdminPassword 🔞 :		
DTRIoOptimized :	optimized	T
WorkerIoOptimized :	optimized	r
UCPAdminUserName 🕖 :	admin	
* InstancePassword 🔞 :		
DTRMaxAmount :	1	
DTRImageId :	ubuntu_14_0405_64_40G_alibase_20170525.vhd	
ControllerIoOptimized :	optimized	T

DDC 访问

使用 ROS 创建 DDC 成功后,您可以进入 ROS 的资源栈管理页面,找到所创建的资源栈,并单击资源栈名称 或单击右侧的**管理**进入资源栈的概要信息页面。

资源编排 ROS	资源栈列表 4	해(1 444)(2	44대 3 4	¥东 1 华东 2	华南 1 雪	港 亚太东北 1 (东京) 亚太东南 1 (新加坡)	亚太东南 2 (悉尼)	美国东部 1 (弗吉尼亚)	6	新建资源栈 🔸	こ別新
资源线管理		調西部 1 (硅谷)	中东东部	1(迪拜) 欧洲	仲部 1 (法兰	冠福)						
资源关型	欢迎加入ROS旺度	王交流群进行讨论	和反馈,旺旺	任群号:149600	6086.							
橫板样例	资源栈名称 ▼	请输入资源栈名	3进行查询		搜索							
▶ 关键帮助	名称			状态 (所有) 👻	超时 (分钟) 失败回调	状态描述		创建的	间		操作
	aliyun_docker_da	atacenter_v1		● 创建完成	60	是	Stack CREATE co	ompleted successfully	2017-	06-27 14:34:18	管理	
										共有1条 ,每页显示:10条	« < <u>1</u>	> >

您可以输出查看登录 UCP 和 DTR 的地址。

在浏览器中输入 UCP 的地址就会显示 UCP 的访问页面,输入在安装 UCP 时创建的管理账号和密码,系统会提示导入 Licence 文件,把准备好的 Licence 导入,即可进入 UCP 的控制界面。



10.在容器服务上轻松搭建 Concourse Cl

Concourse CI 是一款 CI/CD 工具,它的魅力在于极简设计,被广泛应用于 Cloud Foundry 各个模块的 CI/CD。Concourse CI 官方提供了标准的 Docker 镜像,您可以通过阿里云容器服务快速部署一套 Concourse CI 应用。

如果您还不了解 Concourse CI 这款工具,您可以先了解一下 Concourse 的原理。请参见 concourse 官网。

worker worker	worker	worker	final-release
controller monal	controller integration integration-suite	an controller denlor release denlor	

创建 Swarm 集群

登录 <mark>容器服务管理控制台</mark> 创建一个集群。本示例中以包含 1 个节点,网络类型为 VPC 的 swarm 集群为例进 行说明。

关于如何创建集群,请参见创建集群。

⑦ 说明 在创建容器集群时,请保留 EIP。因为需要为 Concourse 配置外部 URL 地址,让您从本机访问 concourse 的 web 服务。

容器服务		集群列表					您最多可	瓜创建 10 ~	↑集群,每个集群最多可!	以添加 50 个节点	刷新	创建Swarm集群	-
Kubernetes	Swarm												
概范		常见问题: ③ 如何创建集群 ③ 如何添	加已有云服务器	の時	可用区节点管理 🔗 集成日志服	务 🖉 通过	Docker客户端道	接集群					
应用		名称 ▼											
服务		集群名称/ID	集群类型	地域	网络类型	集群状态	节点状态 🖉	节点个数	创建时间	Docker版本			操作
集群		test cl	阿里云集群	华东1	虚拟专有网络 vpc- <mark>111111111111111111111111111111111111</mark>	●运行中	健康 ℃	1	2018-05-14 15:43:42	17.06.2-ce	管理	- 査君日志 f 監控 見	删除 更多 ▼

配置安全组规则

Concourse 的组件 ATC 默认监听 8080 端口,因此您需要为集群的安全组配置 8080 端口的入网权限。

- 1. 在 <mark>容器服务管理控制台</mark> 的 swarm 集群列表页面,选择前面创建的集群并单击右侧的**管理**,进入集群详 情页面。
- 2. 在集群的基本信息中,单击安全组的ID,跳转到集群的安全组页面。

集群:test				开启日志服务	登录镜像仓库	刷新
基本信息				升级Agent	升级系统服务	清理磁盘
集群ID: d 1707 150 000 0 0007 0 100770	虚拟专有网络	●运行中	地域: 华东1	节点个数1	集群扩容 添加	加已有节点
安全组ID: sg····································	全组					

3. 单击左侧导航栏中的安全组规则,并单击页面右上角的添加安全组规则。

<	aliclo	o ado oraz	💶 🔷 VPC1 / vpc-					教我设置 😋 返回	添加安全组规则快速	3)建規則 添加Class	sicLink安全	组规则
安全组内实例列表	入方向	出方向							2	土 导入规则	초	部规则
安全组规则	授权策略	协议类型	第口范围	授权类型	授权对象	描述	优先级	创建时间				操作
	允许	全部	-1/-1	地址段访问	172.18.0.0/16		100	2018-01-29 14:49:58		修改描述	克隆	删除
	允许	自定义 TCP	443/443	地址殿访问	0.0.0/0		100	2018-01-29 14:49:57		修改描述	克隆	删除
	允许	全部 ICMP	-1/-1	地址段访问	0.0.0.0/0		100	2018-01-29 14:49:57		修改描述	克隆	2010:
	允许	自定义 TCP	80/80	地址般访问	0.0.0.0/0		100	2018-01-29 14:49:56		修改描述	「売隆」	删除

4. 为安全组配置 8080 端口的入网权限并单击确定。

添加多	R全组规则		? ×
	网卡类型:	内网 ▼	
	规则方向:	入方向	
	授权策略:	允许 🔻	
	协议类型:	自定义 TCP ▼	
	❣ 端□范围:	8080/8080	
	优先级:	1	
	授权类型:	地址段访问	
	* 授权对象:	0.0.0.0/0	🚺 教我设置
	描述:		
		上 长度为2-256个字符,不能以http://或https://开头。	1
		确定	取消

在 ECS 节点上创建 key

为了安全运行 Concourse, 您需要生成 3 个私有 key。

1. 登录到 ECS 节点上,在根目录下,创建目录 keys/web 和 keys/worker。您可以执行以下命令,快速创建这两个目录。

mkdir -p keys/web keys/worker

2. 执行以下命令, 生成 3 个私有 key。

ssh-keygen -t rsa -f tsa_host_key -N '' ssh-keygen -t rsa -f worker_key -N '' ssh-keygen -t rsa -f session_signing_key -N ''

3. 将证书拷贝到相应的目录下。

- cp ./keys/worker/worker_key.pub ./keys/web/authorized_worker_keys
- cp ./keys/web/tsa_host_key.pub ./keys/worker

部署 Concourse Cl

- 1. 登录 容器服务管理控制台。
- 2. 在 Swarm 菜单下,单击左侧导航栏中的 配置项,创建配置文件,变量名称为 CONCOURSE_EXTERNAL_URL,变量值为 http://your-ecs-public-ip:8080 。

*配置文件名:	CONCOURSE_EXTERNAL_URL
	名称长度最大为32个字符,最小为1个字符。
描述:	A
	描述最长不能超过128个字符。
配置项:	编辑配置文件
	变量名称 变量值 操作
	CONCOURSE_EXTERNAL _URL http:// \$18080 编辑 删除
	名称 值 添加
	确定取消

- 3. 单击左侧导航栏的 应用,选择本示例中使用的集群,并单击 创建应用。
- 4. 填写应用的基本信息,并选择使用编排模板创建应用。模板如下。

version: '2' services: concourse-db: image: postgres:9.5 privileged: true environment: POSTGRES_DB: concourse POSTGRES_USER: concourse POSTGRES_PASSWORD: changeme **PGDATA:** /database concourse-web: image: concourse/concourse links: [concourse-db] command: web privileged: true depends_on: [concourse-db] ports: ["8080:8080"] volumes: ["/root/keys/web:/concourse-keys"] restart: unless-stopped # required so that it retries until conocurse-db comes up environment: CONCOURSE_BASIC_AUTH_USERNAME: concourse CONCOURSE_BASIC_AUTH_PASSWORD: changeme CONCOURSE_EXTERNAL_URL: "\${CONCOURSE_EXTERNAL_URL}" CONCOURSE_POSTGRES_HOST: concourse-db CONCOURSE_POSTGRES_USER: concourse CONCOURSE_POSTGRES_PASSWORD: changeme CONCOURSE_POSTGRES_DATABASE: concourse concourse-worker: image: concourse/concourse privileged: true links: [concourse-web] depends_on: [concourse-web] command: worker volumes: ["/keys/worker:/concourse-keys"] environment: CONCOURSE_TSA_HOST: concourse-web dns: 8.8.8.8

5. 单击创建并部署时, 会提示您配置模板参数, 您需要选择关联配置文件, 再单击使用配置文件变量。

模板参数		×
关联配置文件: CONCOURSE_E	XTERNAL_URL V	
参数	值	与配置文件 对比
CONCOURSE_EXTERNAL_URL	http://	相同
与配置文件对比说明: 相同在选择的关联配置文件中,7 不同在选择的关联配置文件中,7 缺失在选择的关联配置文件中,7	有相同的变量名,且变量值相同 有相同的变量名,但变量值不同 殳有相同的变量名	
	使用配置文件变量 确定	定取消

应用创建之后会启动如下 3 个服务。

服务列表	容器列表	日志	事件	路由列表				
服务名称			所属应用	3	服务状态	容器状态	镜像	操作
concourse-w	orker		concour	se-test	●就绪	就绪:1 停止:0	concourse/concourse:latest	停止 重启 重新调度 交更配置 删除 事件
concourse-db	ourse-db concourse-test		● 就绪	就绪:1 停止:0	postgres:9.5	停止 重启 重新调度 交更配置 删除 事件		
concourse-w	eb		concour	se-test	● 就绪	就绪:1 停止:0	concourse/concourse:latest	停止 重启 重新调度 交更配置 删除 事件

至此, Concourse CI部署完成, 在浏览器输入 http://your-ecs-public-ip:8080 即可进行访问。



运行 CI 任务 (Hello world)

- 1. 在上一步浏览器中下载对应您操作系统的 CLI, 然后安装 CLI 客户端,本例以 ECS(Ubuntu16.04)为例。
- 针对 Linux 和 Mac OS X 系统,首先需要给下载的 FLY CLI 文件添加执行权限,然后安装到系统并添加 \$PATH 中:

chmod +x fly install fly /usr/local/bin/fly

3. 安装之后可以查看一下版本。

\$fly -v 3.4.0

4. 连接 Target,用户名和密码默认是 concourse 和 changeme。

```
$ fly -t lite login -c http://your-ecs-public-ip:8080
in to team 'main'
username: concourse
password:
saved
```

5. 将下面的配置模板保存为 hello.yml 。

```
jobs:

- name: hello-world

plan:

- task: say-hello

config:

platform: linux

image_resource:

type: docker-image

source: {repository: ubuntu}

run:

path: echo

args: ["Hello, world!"]
```

6. 注册任务。

fly -t lite set-pipeline -p hello-world -c hello.yml

7. 启动任务。

fly -t lite unpause-pipeline -p hello-world

成功执行的界面如下。

≡ *
hello-world #1 started 4h 6m ago duration 28s
1
>_ say-hello
<pre>Pulling ubuntu@sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad: Pulling from library/ubuntu d5c6f90da05d: Pulling fs layer c220aa3cfclb: Pulling fs layer dc27a084064f: Waiting dc27a084064f: Waiting c220aa3cfclb: Verifying Checksum c220aa3cfclb: Verifying Checksum 1300883d87d5: Pulling Checksum 1300883d87d5: Verifying Checksum 1300883d87d5: Verifying Checksum 2e9398f099dc: Verifying Checksum 2e9398f099dc: Verifying Checksum 2e9398f099dc: Verifying Checksum 1300883d87d5: Download complete dc27a084064f: Waiting 2e9398f099dc: Verifying Checksum 1300883d87d5: Pullicomplete 2e9398f099dc: Verifying Checksum 2e9398f099dc: Verifying Checksum 2e9398f099dc: Verifying Checksum 2e9398f099dc: Pullicomplete d5c6f90da05d: Verifying Checksum d5c6f90da05d: Verifying Checksum d5c6f90da05d: Pullicomplete d226084Cfclb: Pullicomplete 2e9398f099dc: Pullicomplete d22708406ff: Pullicomplete d22708406f</pre>
Status: Downloaded newer image for ubuntugsha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad
Successfully pulled ubuntu#shs256;34471448724419596cs4e890496d375801de21b0e67b81s77fd6155ce001edad. Hello, world!

更多关于 Concourse CI 的特性,请参见 Concourse CI 项目。

11.利用Terraform部署Swarm集群及 Wordpress应用

本文档介绍了如何通过 Terraform 在 VPC 环境下部署一个阿里云容器服务集群,并在该集群之上,部署一个 WordPress 样例应用。本文档提供一种构建阿里云基础设施的解决方案,让您通过代码来自动创建、编排和 管理容器服务以及完成在容器集群上应用的自动化部署。

前提条件

- 需要开通阿里云容器服务,用户的账户需有100元余额并通过实名认证。
- 需要启用用户账号的 AccessKey, 妥善保存和记录 AccessKey ID 和 AccessKey Secret。

步骤 1 安装 Terraform

下载 Terraform

在 Terraform 官方下载地址 下载 Terraform您可以选择合适的版本和平台。本文档以在 Linux 上安装 Terraform 为例(操作步骤与 Mac OS X 平台十分相似)。

- 1. 单击 Linux 图标下载 terraform_0.11.3_linux_amd64.zip 文件。
- 2. 复制该 .zip 文件到合适的路径中,本例中为 /usr/local/terraform。
- 3. 解压缩该文件, 您会得到一个二进制文件 terraform。
- 4. 在 */etc/profile*下创建以下条目,将二进制文件所在的路径 */usr/local/terraform*添加到 PATH环境变量中。

export TERRAFORM_HOME=/usr/local/terraform export PATH=\$PATH:\$TERRAFORM_HOME

安装阿里云 Terraform Provider

在使用Terraform之前,需要进行初始化操作来加载面向阿里云的Provider。在模板文件目录下运行命令:

terraform init

加载成功后,将会把相应的 Plugin 下载到当前文件夹下的 *.terraform* 隐藏目录下。如果在加载过程中遇到了 网络超时的问题,可按照接下来的步骤完成插件的手动安装。

- 在阿里云 Terraform Provider 官方下载地址下载对应版本和平台的 Provider。本例中选择 Linux 类型。
- 复制下载的文件 *terraform-provider-alicloud_1.9.3_linux_amd64.zip*到 Terraform 的安装目录 /usr/local /terraform并解压即可。将解压后,当前目录将会得到阿里云的 Provider *terraform-provider-alicloud_v1*.9.3_x4。

Terraform 和 Provider 安装成功后,运行以下命令检测 Terraform 的运行,若成功安装,应显示以下内容。

\$ terraform

Usage: terraform [--version] [--help] [args] The available commands for execution are listed below. The most common, useful commands are shown first, followed by less common or more advanced commands. If you're just getting started with Terraform, stick with the common commands. For the other commands, please read the help and docs before usage. Common commands:

All other commands: debug Debug output management (experimental) force-unlock Manually unlock the terraform state state Advanced state management

步骤2 下载容器服务 Swarm 和应用 Wordpress 的 Terraform 模板

您可以从 Git Hub上下载创建 Swarm 集群和部署 Wordpress 应用的 Terraform 模板(模板下载地址)。该模 板文件定义了创建 swarm 集群的相关资源以及在 swarm 集群上部署 Wordpess 的文件,帮助您完成 Swarm 集群的快速创建和应用的快速部署。 模板中包含以下文件。

main.tf

Terraform 主文件。定义了将要部署的资源。

● 地域

定义了资源将要被创建在哪个地域里。

```
provider "alicloud" {
  access_key = "${var.alicloud_access_key}"
  secret_key = "${var.alicloud_secret_key}"
  region = "${var.region}"
}
```

```
    VPC
```

```
resource "alicloud_vpc" "vpc" {
name = "${var.vpc_name}"
cidr_block = "${var.vpc_cidr}"
}
```

VSwitch

```
resource "alicloud_vswitch" "vswitch" {
  availability_zone = "${data.alicloud_zones.default.zones.0.id}"
  name = "${var.vswitch_name}"
  cidr_block = "${var.vswitch_cidr}"
  vpc_id = "${alicloud_vpc.vpc.id}"
}
```

容器服务集群

```
resource "alicloud_cs_swarm" "cs_vpc" {
  password = "${var.password}"
  instance_type = "${data.alicloud_instance_types.main.instance_types.0.id}"
  name = "${var.cluster_name}"
  node_number = "${var.node_number}"
  disk_category = "${var.node_number}"
  disk_size = "${var.disk_category}"
  disk_size = "${var.disk_size}"
  cidr_block = "${var.cidr_block}"
  image_id = "${data.alicloud_images.main.images.0.id}"
  yswitch_id = "${alicloud_vswitch.main.id}"
}
```

• Wordpress 应用

```
resource "alicloud_cs_application" "wordpress" {
  cluster_name = "${alicloud_cs_swarm.cs_vpc.name}"
  name = "${var.app_name == "" ? var.resource_group_name : var.app_name}"
  version = "${var.app_version}"
  template = "${file("wordpress.yml")}"
  description = "terraform deploy consource"
  latest_image = "${var.latest_image}"
  blue_green = "${var.blue_green}"
  blue_green_confirm = "${var.confirm_blue_green}"
}
```

outputs.tf

该文件定义了输出参数。作为执行的一部分而创建的资源会生成这些输出参数。和 ROS 模板指定的输出参数类似。例如,该模板将部署一个 Swarm 集群和 Wordpress 应用实例。以下输出参数将提供集群 ID 和应用的默认域名。

```
output "cluster_id" {
  value = "${alicloud_cs_swarm.cs_vpc.id}"
}
```

```
output "default_domain" {
  value = "${alicloud_cs_application.wordpress.default_domain}"
}
```

variables.tf

该文件包含可传递到 main.tf 的变量, 可帮助您自定义环境。

```
variable "alicloud_access_key" {
    description = "The Alicloud Access Key ID to launch resources. Support to environment 'ALICLOUD_ACCESS
_KEY'."
}
```

```
variable "alicloud_secret_key" {
    description = "The Alicloud Access Secret Key to launch resources. Support to environment 'ALICLOUD_SE
    CRET_KEY'."
}
```

```
variable "region" {
  description = "The region to launch resources."
  default = "cn-hongkong"
}
```

```
variable "vpc_cidr" {
  description = "The cidr block used to launch a new vpc."
  default = "172.16.0.0/12"
}
```

```
variable "app_name" {
  description = "The app resource name. Default to variable `resource_group_name`"
  default = "wordpress"
}
```

wordpress.yml

部署 Wordpress 应用的 Compose 模板,该模板来自于控制台提供的编排模板,详情登录容器服务控制台, 单击左侧导航栏中的**应用**,选择**创建应用 > 使用编排模板创建 > 使用已有编排模板**。

步骤 3 执行 Terraform 脚本

要想运行脚本,首先定位到您存放以上文件的目录,如 /root/terraform/wordpress。您可以利用以下 terraform 的相关命令,运行脚本,构建容器集群和部署应用。更多的命令用法,参见 Terraform Commands (CLI)。

```
执行 terraform init , 会初始化环境。
```

\$ terraform init

```
Initializing provider plugins...
```

•••

- Checking for available provider plugins on https://releases.hashicorp.com...

- Downloading plugin for provider "alicloud" (1.7.2)...

* provider.alicloud: version = "~> 1.7"

Terraform has been successfully initialized!

•••

执行 terraform providers , 会列出安装的供应商。

terraform providers

— provider.alicloud

在执行 terraform plan 之前,首先需要传递 AccessKey ID 和 AccessKey Secret 进行授权。

```
$ export ALICLOUD_ACCESS_KEY="AccessKey ID"
$ export ALICLOUD_SECRET_KEY="AccessKey Secret"
```

然后执行 terraform plan , 会创建一个执行计划, 并帮助您了解将要创建或改变的资源。

can't guarantee that exactly these actions will be performed if "terraform apply" is subsequently run.

确定资源已如您所愿地创建/更新后,运行 terraform apply 命令,开始执行 Terraform 模块。

\$ terraform apply data.alicloud_instance_types.default: Refreshing state... data.alicloud_images.main: Refreshing state... data.alicloud_zones.default: Refreshing state... An execution plan has been generated and is shown below. Resource actions are indicated with the following symbols: + create Terraform will perform the following actions: Plan: 9 to add, 0 to change, 0 to destroy. Do you want to perform these actions? Terraform will perform the actions described above. Only 'yes' will be accepted to approve. Enter a value: yes alicloud_vpc.vpc: Creating... Apply complete! Resources: 9 added, 0 changed, 0 destroyed. Outputs: ##注意 availability_zone = cn-hongkong-a cluster_id = c95537435b******* default_domain = c95537435b*******.cn-hongkong.alicontainer.com vpc_id = vpc-2zeaudqan6uzt5lzry48a vswitch_id = vsw-2ze2x92n9b5neor7fcjmr

terraform apply 命令执行完毕后, 会显示 outputs.tf 中定义的输出参数。在上面这个例子中, 输出参数 为 cs_cluster 集群 ID、可用区、VPC ID、VSwitch ID 名称和应用实例的 def ault_domain。

通过运行 terraform output 命令,可随时查看输出值,帮助您配置 WordPress 应用。
terraform output availability_zone = cn-hongkong-a cluster_id = c95537435b******* default_domain = c95537435b*******.cn-hongkong.alicontainer.com vpc_id = vpc-2zeaudqan6uzt5lzry48a vswitch_id = vsw-2ze2x92n9b5neor7fcjmr

您现在可以在容器服务控制台查看通过 t erraform 创建的集群,查看集群信息、节点信息、日志信息和容器 信息等信息。

容器服务 - Swarm ▼	集群列表				42	最多可以创建	115 个集群	,每个集群最多可以涨	劾 40 个节点	刷新	创建Swarm集群	•
概览	常见问题: & 如何创建集群 & 如何]添加已有云服务器	番 🔗 跨可	「用区节点管理 🔗 集成日志!	69 🔗 🖉 进	过Docker客所	中端连接集	Ħ				
应用	名称 ▼											
	集群名称/ID	地 集群类型 ▼	域 (全部)	网络类型	集群状 态	节点状态 ②	节点个 数	创建时间	Docker版 本		2	操作
节点	test	阿里云集 群	东1	虚拟专有网络 vpc- bp1kd7yn4qnr8ganuevq5	●运行 中	健康 🕽	1	2018-06-25 17:01:45	17.06.2- ce	管理	查看日志 监控	删除 更多 ▼

同时,可以在应用页面查看 Wordpress 应用信息。

容器服务 - Swarm ▼	应用列表				刷新创建应用
概览	常见问题 2.(何创建应用 & 变更应用配置 & 简	单路由蓝绿发布策略 🔗 容器弹性伸缩			
应用 1	集群: 🛛 🔹 🦉 隐藏系统应用 📄 隐藏离线应用	□ 隐藏在线应用		名称 ▼	Q X
服务	应用名称 描述 状态	容器状态 创建时间 🔺	更新时间 ▲		撮作
集群 节点	wordpress 3 ●就编	就结:0 2018-07-06 18:17:19 停止:0	2018-07-06 18:17:19	停止 🥼 变更配置	删除 重新部署 事件

单击应用名称,然后单击**路由列表**,可查看路由地址。

触发器 1	競波器 1.每种类型的競发器只能创建1个◎					
目前没有任何	「触发器,点击右	i上角按钮(创建触发器	ł		
服务列表	容器列表	日志	事件	路由列表		
路由地址 (路由地址(集群鄉定SLB后路由地址才能被访问)					
wordpress.	wordpress.					

步骤 4 访问 WordPressPress

- 1. 打开 Wordpress Compose 模板 wordpress.yml , 找到应用域名前缀 aliyun.routing.port_80: http://w ordpress 。
- 将域名前缀 http://wordpress 与应用的 default_domain 拼接后的值 http://wordpress.c95537435b**
 ******.cn-hongkong.alicontainer.com 输入浏览器,即可访问 WordPress 欢迎页面,可选择语言,然后 继续配置。



3. 输入站点名称以及管理员的用户名和密码。选择安装 WordPress。

Below you should en host.	ter your database connection de	etails. If you're not sure about these, contact your
Database Name	wp_db	The name of the database you want to use with WordPress.
Username	wp_admin	Your database username.
Password	Test12345	Your database password.
Database Host	he bedgall (Hawiter)	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	wp_	If you want to run multiple WordPress installations in a single database, change this.
Submit		

4. WordPress 安装完成后,单击登录,输入管理员的用户名和密码,进入 WordPress 应用。

) 😤 Test 🛡 0 ·	+ New			Howdy, admir			
Dashboard	Dashboard			Screen Options * Help *			
me							
dates	Welcome to WordPress!			O Dismiss			
Posts	We've assembled some links to get you starte	ed:					
Media	Get Started	Next Steps	More Actions	More Actions			
Pages		Write your first blog post	Hanage widgets or	Manage widgets or menus			
Comments	Customize Your Site	+ Add an About page	Turn comments on or off				
Annearance	or, change your theme completely	View your site	Learn more about getting started				
Plugins							
Users	At a Glance	A Quick Draft	t				
Tools	🖈 1 Post 📕 1 Page	Title					
Settings	I Comment	What's on	your mind?				
Collapse menu	WordPress 4.9.2 running Twenty Seventeen theme.	What's on	What's on your minut				
	Activity	Save Draft					
	Recently Published						
	Preset Comments	WordPress	Events and News				
	Recent Comments	Attend an up	ocoming event near you. 🖉				
	From A WordPress Commenter on Hello world! Hi, this is a comment. To get started with mode comments, please visit the Comments screen in	rating, editing, and deleting	Working Day dge, England, United Kingdom	Tuesday, Feb 6, 201 8:00 an			
		wordPr	ress Maidstone Meetup	Thursday, Feb 8, 201			

延伸阅读

阿里云目前是 Terraform 官方的 Major Cloud Provider,如果您想通过 terraform 灵活构建阿里云上的基础 设施资源,您可参见 Alicloud Provider 了解更多信息,自定义资源描述文件,快速搭建属于您的云上设施。

12.Chef实现Dokcer和WebServer自动 化部署

Chef 是一个自动化部署框架,结合阿里云容器服务,可实现定制化的自动部署工作。首先参考Chef 官网了 解一些基本概念,诸如 Cookbook、Recipe、Chef Workstation、Chef Server、Chef Nodes,以便快速入 门。

前提条件

- 您已成功创建一个Swarm集群,该集群保留EIP。
- 准备一个本地Linux环境,本示例是Ubuntu 16.04,您可根据本地环境,参 考https://downloads.chef.io/chefdk/获取相应的ChefDK。
- 您需要登录Chef官网,注册一个账号,创建一个Organization,本例中为example。

在Linux上安装chef工作站

您需要前往chef官网下载符合本地Linux环境的ChefDK,本例中使用Ubuntu 16.04对应的ChefDK。

首先在/home目录下创建一个chef-repo文件夹。

mkdir /home/chef-repo

进入chef-repo目录,使用curl命令下载ChefDK程序包,并进行安装。

cd /home/chef-repo curl -O https://packages.chef.io/files/stable/chefdk/3.0.36/ubuntu/16.04/chefdk_3.0.36-1_amd64.deb dpkg -i chefdk_3.0.36-1_amd64.deb

然后您需要进行大量的Chef安装配置,您在安装过程中如果遇到问题,可参见Chef官方文档进行排查。

验证Chef

chef verify	#验证ChefDK的组件是否正常
chefversion	#查看chef版本

设置Chef 环境变量

设置Chef相关的环境变量,如:GEM_ROOT、GEM_HOME、GEM_PATH。

export GEM_ROOT="/opt/chefdk/embedded/lib/ruby/gems/2.1.0" export GEM_HOME="/root/.chefdk/gem/ruby/2.1.0" export GEM_PATH="/root/.chefdk/gem/ruby/2.1.0:/opt/chefdk/embedded/lib/ruby/gems/2.1.0"

此外,如果你的系统上已经安装了ruby,你需要更新与ruby相关的PATH变量。

export PATH="/opt/chefdk/bin:/root/.chefdk/gem/ruby/2.1.0/bin:/opt/chefdk/embedded/bin:/opt/chefdk/bi n:/root/.chefdk/gem/ruby/2.1.0/bin:/opt/chefdk/embedded/bin:/opt/chefdk/bin:/root/.chefdk/gem/ruby/2.1. 0/bin:/opt/chefdk/embedded/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/root/bin"

设置访问Chef的Firewalld规则

为了访问Chef服务器上的Chef Manage GUI,添加以下firewalld规则,开放Chef服务器上的相应端口。

```
firewall-cmd --direct --add-rule ipv4 \
filter INPUT_direct 0 -i eth0 -p tcp \
--dport 443 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 \
filter INPUT_direct 0 -i eth0 -p tcp \
--dport 80 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 \
filter INPUT_direct 0 -i eth0 -p tcp \
--dport 9683 -j ACCEPT
firewall-cmd --reload
```

从Chef Manage GUI下载Starter Kit

登录Chef Manage GUI,单击Administration选项,从列表中选择organization。此例中,organization为 example,选中organization之后,点击左侧菜单中的Starter Kit,将chef-starter.zip文件下载到本地机器。

将chef-starter.zip文件传输到本地Linux下的Chef工作站,并解压到home/chef-repo目录下。

cd /home/chef-repo unzip chef-starter.zip

下载Chef服务器的SSL证书

证书会下载到chef-repo/.chef/trusted_certs目录中。

cd ~/chef-repo # knife ssl fetch WARNING: Certificates from api.chef.io will be fetched and placed in your trusted_cert directory (/root/chef-repo/.chef/trusted_certs). Knife has no means to verify these are the correct certificates. You should verify the authenticity of these certificates after downloading. Adding certificate for wildcard_opscode_com in /root/chef-repo/.chef/trusted_certs/wildcard_opscode_co m.crt Adding certificate for DigiCert_SHA2_Secure_Server_CA in /root/chef-repo/.chef/trusted_certs/DigiCert_SH A2_Secure_Server_CA.crt

验证Chef工作站是否安装成功

配置成功后,执行以下命令,可看到我们预先创建的Organization,说明我们已成功连接到工作站。

cd ~/chef-repo
knife client list
example-validator

创建实现Docker自动初始化的CookBook

- 1. 在Chef工作站上创建一个CookBook。
 - 在chef-repo/cookbooks目录下,执行以下命令,创建一个名为docker_init的CookBook。

chef generate cookbook docker_init

 ○ 进入*chef-repo/cookbooks/docker_init/recipe/*目录,找到def ault.rb文件,进行配置。该示例用于 在Ubunt u中启动最新的Docker版本。

apt_update package 'apt-transport-https' package 'ca-certificates' package 'curl' package 'software-properties-common' execute 'apt-key' do command 'apt-key fingerprint 0EBFCD88' end execute 'apt-repo' do command 'add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu/dists/ xenial/stable/" end execute 'apt-repo' do command 'apt-get update' end execute 'apt-repo' do command 'apt-get install docker-ce -y --allow-unauthenticated' end service 'docker' do action [:start, :enable] end

2. 校验docker_init这个CookBook是否在本地工作。

chef-client --local-mode --runlist 'recipe[docker_init]' [2018-06-27T15:54:30+08:00] INFO: Started chef-zero at chefzero://localhost:1 with repository at /root/c hef-repo One version per cookbook Starting Chef Client, version 14.1.12 [2018-06-27T15:54:30+08:00] INFO: *** Chef 14.1.12 *** [2018-06-27T15:54:30+08:00] INFO: Platform: x86_64-linux [2018-06-27T15:54:30+08:00] INFO: Chef-client pid: 2010 [2018-06-27T15:54:30+08:00] INFO: The plugin path /etc/chef/ohai/plugins does not exist. Skipping... [2018-06-27T15:54:31+08:00] INFO: Setting the run_list to [#] from CLI options [2018-06-27T15:54:32+08:00] INFO: Run List is [recipe[docker_init]] [2018-06-27T15:54:32+08:00] INFO: Run List expands to [docker_init] [2018-06-27T15:54:32+08:00] INFO: Starting Chef Run for yxm [2018-06-27T15:54:32+08:00] INFO: Running start handlers [2018-06-27T15:54:32+08:00] INFO: Start handlers complete. resolving cookbooks for run list: ["docker init"] [2018-06-27T15:54:32+08:00] INFO: Loading cookbooks [docker_init@0.1.0] Synchronizing Cookbooks: - docker_init (0.1.0) Installing Cookbook Gems: Compiling Cookbooks... **Converging 10 resources** Recipe: docker_init::default * apt_update[] action periodic[2018-06-27T15:54:32+08:00] INFO: Processing apt_update[] action perio dic (docker_init::default line 9) •••• ---- End output of add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu/d ists/xenial/stable/" ----Ran add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu/dists/xenial/st

able/" returned 1

执行以下命令,检查本地安装的Docker是否升级到最新版本。

docker --version Docker version 17.06.2-ce, build 2e0fd6f

- 3. 将该CookBook上传到Chef Server。
 - 在Chef工作站,执行以下命令,将名为docker_init的CookBook上传到Chef Server。

knife cookbook upload docker_init

○ 执行以下命令,检验该CookBook是否被成功上传。

knife cookbook list
docker_init 0.1.0

- 4. 将该cookbook导入到阿里云Swarm集群的节点。
 - 在Chef工作站执行以下命令,将docker_init导入充当Chef Node的Swarm集群的节点。

(?) 说明 将ADDRESS替换为Swarm集群的ECS节点的EIP, USER是ECS节点登录用户, 一般为 root, PASSWORD是ECS节点登录密码。若Swarm集群中有多个节点, 需要对每个ECS节点执行 该命令。

knife bootstrap ADDRESS --ssh-user USER --ssh-password 'PASSWORD' --sudo --use-sudo-passwor d --node-name node1-ubuntu --run-list 'recipe[docker_init]' Creating new client for node1-ubuntu Creating new node for node1-ubuntu Connecting to 121.196.219.18

https://download.docker.com/linux/ubuntu/dists/xenial/stable/" ----121.196.219.18 Ran add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubun tu/dists/xenial/stable/" returned 1

- 登录每个ECS节点,检查每个节点上安装的Docker是否已更新到最新版本。执行 docker -- version 命 令进行检验。
- 至此,通过Chef自动部署系统,实现对阿里云容器集群Docker的版本更新。

创建自动化部署Web Server的CookBook

- 1. 在Chef工作站创建一个新的CookBook。
 - 在chef-repo/cookbooks目录下,执行以下命令,创建名为web_init的CookBook。

chef generate cookbook web_init

○ 进入chef-repo/cookbooks/web_init/recipe/目录,找到default.rb文件,进行配置。

```
execute 'apt-repo' do
command 'apt-get -y install apache2 --allow-unauthenticated'
end
service 'apache2' do
action [:start, :enable]
end
file '/var/www/html/index.html' do
content '
hello world
'
end
service 'iptables' do
action :stop
end
```

- 2. 检验该CookBook是否在本地工作。
 - 执行 curl http://localhost:80 命令,检查web_init是否在本机工作。
 - 在Chef 工作站,将web_init这个CookBook上传到Chef Server。

knife cookbook upload web_init

3. 将该cookbook导入到阿里云Swarm集群的节点。

在Chef工作站执行以下命令,将web_init导入充当Chef Node的Swarm集群的节点。

⑦ 说明 将ADDRESS替换为Swarm集群的ECS节点的EIP, USER是ECS节点登录用户,一般为 root, PASSWORD是ECS节点登录密码。若Swarm集群中有多个节点,需要对每个ECS节点执行该命 令。 knife bootstrap ADDRESS --ssh-user USER --ssh-password 'PASSWORD' --sudo --use-sudo-password --n ode-name node1-ubuntu --run-list 'recipe[web_init]'

- 4. 在阿里云Swarm集群中,检查Web Server是否成功启动。登录阿里云Swarm集群的节点。
 - 执行 systemctl status apache2.service ,检查apache2 是否正常运行。
 - 在浏览器中访问 http://ADDRESS:80 , 查看浏览器屏幕是否输出 hello world 。

⑦ 说明 ADDRESS是指节点的EIP。