Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Comments |
| 1.0 | kacai | 9/6/2018 | install K8S on azs (global subscription, with in China FW) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

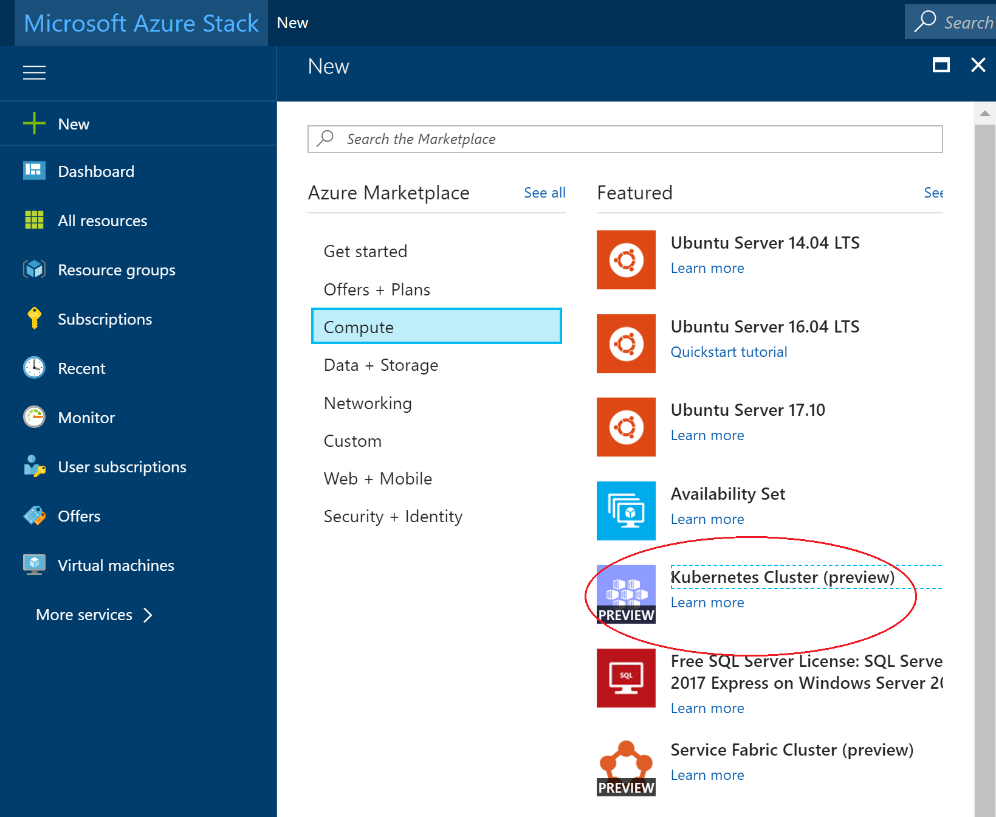
**说明：**

文档中在Azure Stack上安装K8S cluster的步骤，在Azure Stack 1807版本、连接global Azure、Azure Stack处于国内网络环境的情形下验证通过。

在其它情形下，可能需要适当调整。

# Azure Stack支持Kubernetes的现状

Kubernetes Cluster是Azure Stack plan的标准feature，在Marketplace -> Compute下可以找到Kubernetes Cluster服务：



官方给出了安装步骤：<https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-template-kubernetes-deploy>

但是目前有几个问题：

* Kubernets Cluster在Azure Stack中还是private preview，在安装过程中可能经常会遇到问题。而我们能控制的配置项相对比较少，往往只能找产品组协助解决
* Kubernetes Cluster的安装默认行为去从海外站点拉取Kubernetes服务所需的各种镜像。因为国内防火墙的限制，这往往使得不少镜像拉取不成功，导致Kubernetes服务进程不能启动，整个部署失败。直接使用Kubernetes Cluster进行K8S集群部署，很可能会遇到防火墙问题，而我们能用的配置项较少，很难对Kubernetes cluster服务的镜像源地址进行控制

因此，目前阶段在国内网络环境下，如果想在Azure Stack上成功部署Kubernetes集群，使用acs-engine应该是比较可行的办法。和使用acs-engine在Azure公有云上部署K8S集群相比，在Azure Stack上部署K8S集群步骤相对多一点，主要是因为每一个Azure Stack的安装instance，它们的内部profile信息不相同。因此在创建部署K8S集群部署的apiModel文件时，必须引入Azure stack的profile内容，而对于Azure公有云来说这部分是不用反应在apiModel中的。

在Azure Stack上部署K8S并不算困难。下面我们就开始部署K8S on Azure Stack！

# 前期准备

* 确认Azure stack需要在1803版本以上（否则可能的问题，请参考前人的总结：<https://mp.weixin.qq.com/s?__biz=MzA3NTM1MzE4Nw==&mid=2649617746&idx=1&sn=8d110646120244bbab6b55e252fc744e&chksm=8768885eb01f014824a130cb4148cd9c8648e20d155549a1d09df7d88b3dceeec64821ee09dd#rd>）
* Azure Stack与公有云Azure连接
* 确保Azure Marketplace中包含：
  + Publisher = "Canonical" Offer = "UbuntuServer" SKU = "16.04-LTS" OSType = "Linux"
  + Custom Script for Linux, 2.0.3
* 从Azure Stack管理员处获取Azure Stack ERCS访问信息（IP、用户名/密码）
* git clone <https://github.com/kylercai/K8SOnAzureStack.git>
* 创建SSH密钥对，用于SSH访问K8S集群的节点
* 运行acs-engine的Linux VM

# 创建apiModel

* 从命令”git clone <https://github.com/kylercai/K8SOnAzureStack.git>”，可以获取两个关键文件：
  + azurestack-default.json
  + AzureStack.AcsEngine.psm1，我们将要运行这个powershell脚本

**说明：**

AzureStack.AcsEngine.psm1脚本的原始版本可以从<https://github.com/radhikagupta5/AzureStack-QuickStart-Templates/tree/radhikgu-acs/101-acsengine-kubernetes-1803> 找到。经过多次试验后，对原始的版本进行了简化，去掉了创建Service Principal和授权，创建存储账户和上传apiModel等步骤。这些步骤并不是必须的，且比较容易导致访问权限、api版本不兼容等问题，导致大量的troubleshooting要做。去掉这些步骤后，在后续步骤中采用少量的手工操作进行替代，避免原始版本可能导致的困难的troubleshooting。

* 启动Powershell ISE，建议以管理员身份启动
* 依次执行以下命令，并用合适的内容替代highlight部分

Import-Module *d:\MSFT\Azure-Study\AKS\azurestack\mtc\AzureStack.AcsEngine.psm1* -Force -Verbose **// AzureStack.AcsEngine.psm1在本地的完整路径**

$namingSuffix = *23456* **// 希望在K8S master节点的DNS前缀**

$masterDnsPrefix = "k8s-" + $namingSuffix

$tenantSubscriptionId = "*f3bd3-……-bc867b*" **// 当前使用Azure Stack的subscription**

$CloudAdminPass = ConvertTo-SecureString "*xxxxxx*" -AsPlainText -Force **// 访问ERCS的密码**

$cloudAdminCredential = New-Object System.Management.Automation.PSCredential ("*xxxxxxx*", $CloudAdminPass) **// 访问ERCS系统的用户名**

$serviceAdmin = " *xxx@xxxx.xxxx.xxx* " **// 当前访问AzureStack的用户名**

$AdminPass = ConvertTo-SecureString "*xxxxxxx*" -AsPlainText -Force **// 当前访问AzureStack的密码**

$serviceAdminCredential = New-Object System.Management.Automation.PSCredential ($serviceAdmin, $AdminPass)

$TenantAdmin = "*xxx@xxxx.xxxx.xxx*" **// 当前访问AzureStack的用户名**

$TenantAdminPass = ConvertTo-SecureString "*xxxxxxx*" -AsPlainText -Force **// 当前访问AzureStack的密码**

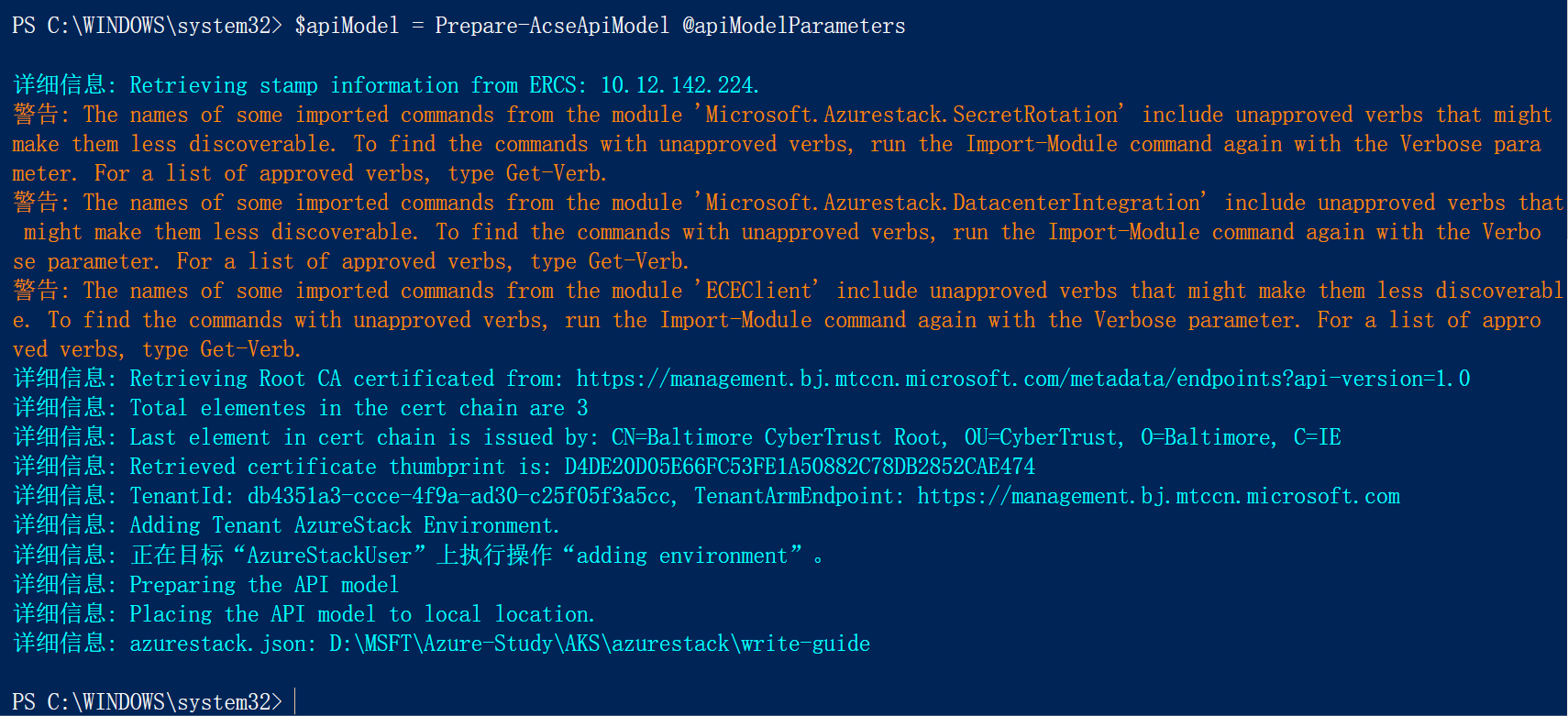
$tenantAdminCredential = New-Object System.Management.Automation.PSCredential ($TenantAdmin, $TenantAdminPass)

$acsSshKey = "*ssh-rsa AAAAB3….voAoeJQw/af azureuser@azconsole*" **// 创建的SSH密钥对的公钥内容**

$apiModelParameters = @{'ErcsComputerName' = "*10.x.x.x*"; 'CloudAdminCredential' = $cloudAdminCredential; 'ServiceAdminCredential' = $serviceAdminCredential; 'TenantAdminCredential' = $tenantAdminCredential; 'TenantSubscriptionId' = $tenantSubscriptionId; 'MasterDnsPrefix' = $masterDnsPrefix; 'LinuxVmSshKey' = $acsSshKey; 'NamingSuffix' = $namingSuffix;} **// ERCS的IP地址**

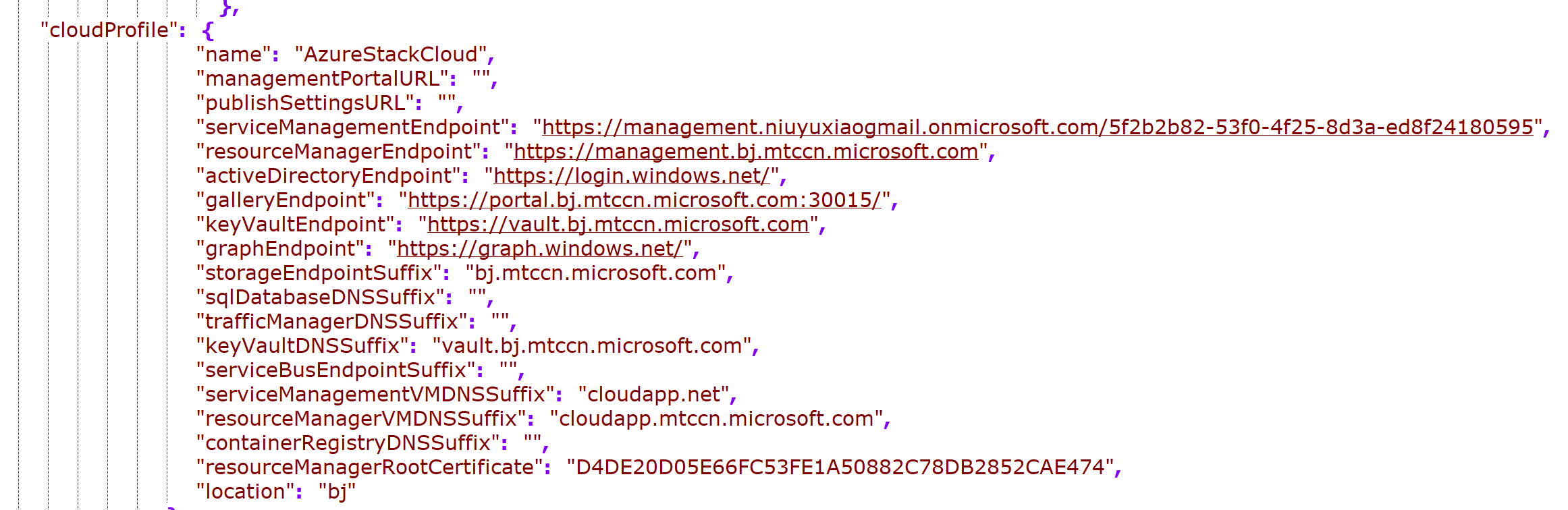
$apiModel = Prepare-AcseApiModel @apiModelParameters

* 如果顺利，执行的结果如下，在红色方框内，显示生成的apiModel文件（azurestack.json）在本地的目录路径：



**说明：**

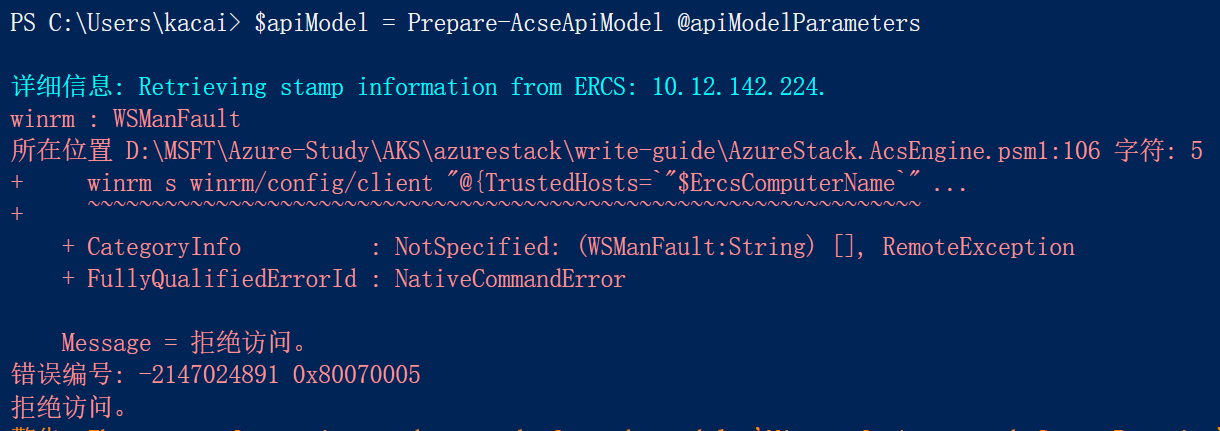
执行过程连接Azure Stack的ERCS，从中抽取Azure Stack Instance的thumbprint、tenantId、endpoint等信息。这些信息被填到生成的apiModel文件中，相应的在cloudProfile部分，大致内容类似：



因为每一个Azure Stack的安装instance，它们的内部profile信息不相同，所以在创建部署K8S集群部署的apiModel文件时，必须引入Azure stack的profile内容，而对于Azure公有云来说这部分是不用在apiModel中的。

**Tip：**

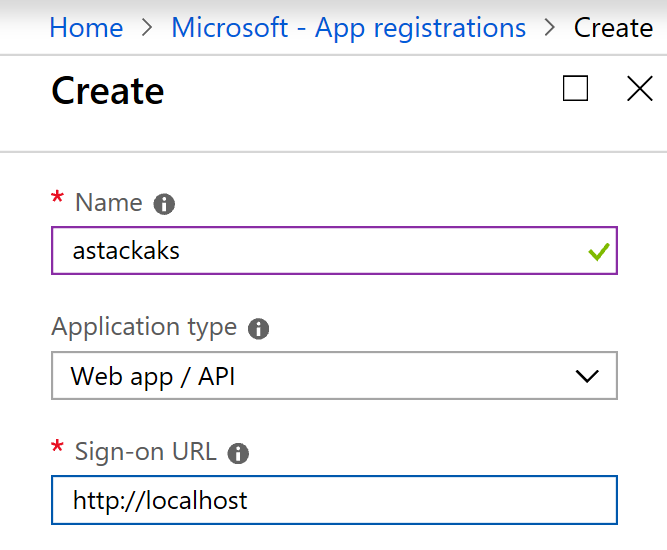
* 建议使用管理员身份启动Powershell ISE来运行以上命令，否则可能会出现本地创建TrustedHosts被拒绝



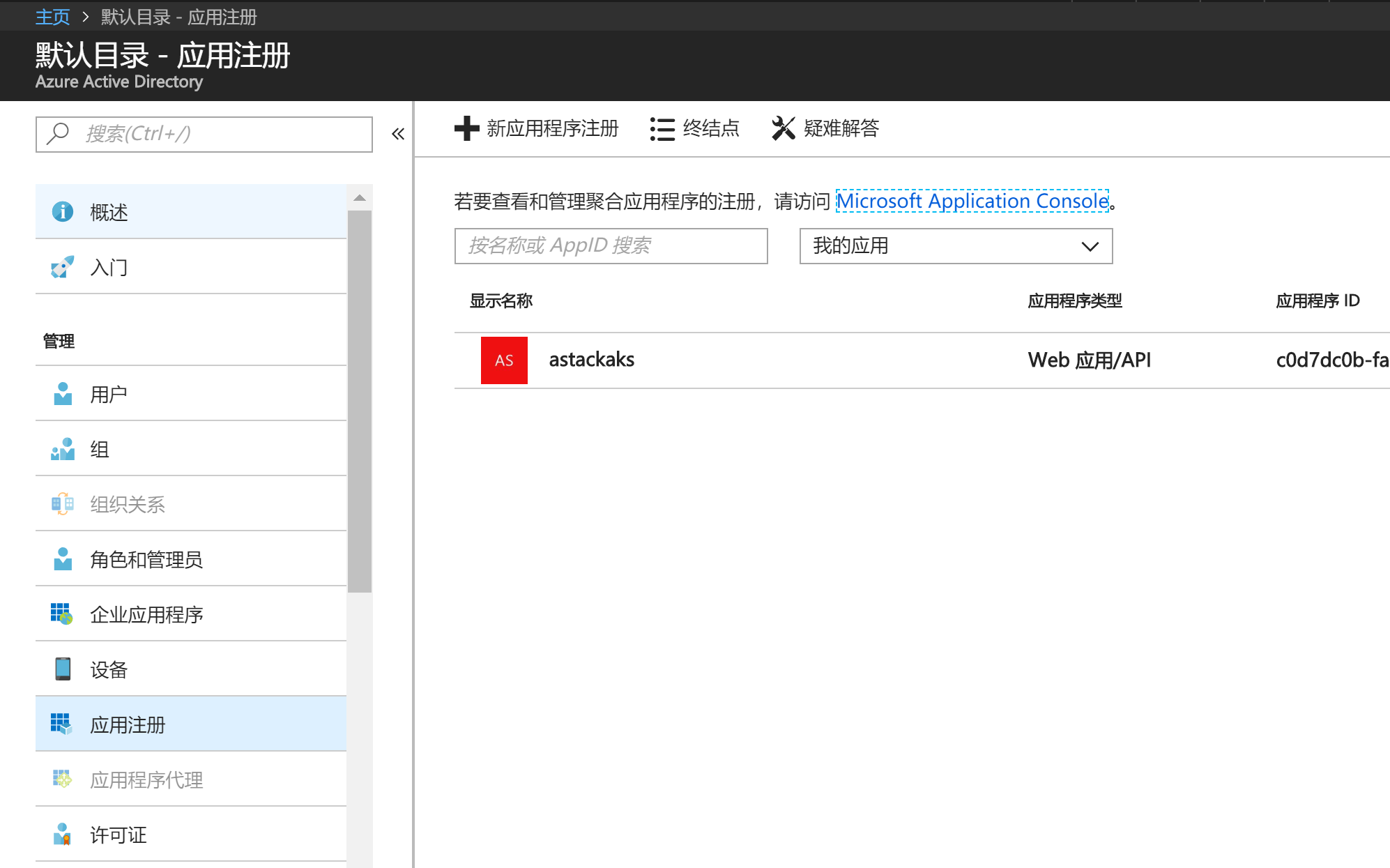
* 在脚本执行过程中，可能会遇到一些问题，但我们的目标只是生成apiModel文件，因此只要脚本还能继续运行下去，最终生成apiModel文件，我们不一定都需要处理所有的问题。<https://github.com/kylercai/K8SOnAzureStack.git> 这个地址下有一个创建成功生成的apiModel文件（azurestack.json）例子，只要您生成apiModel文件和例子对比内容都完整，就不需要关注脚本运行过程中遇到的和我们目标无关的问题。

# 创建Service Principal

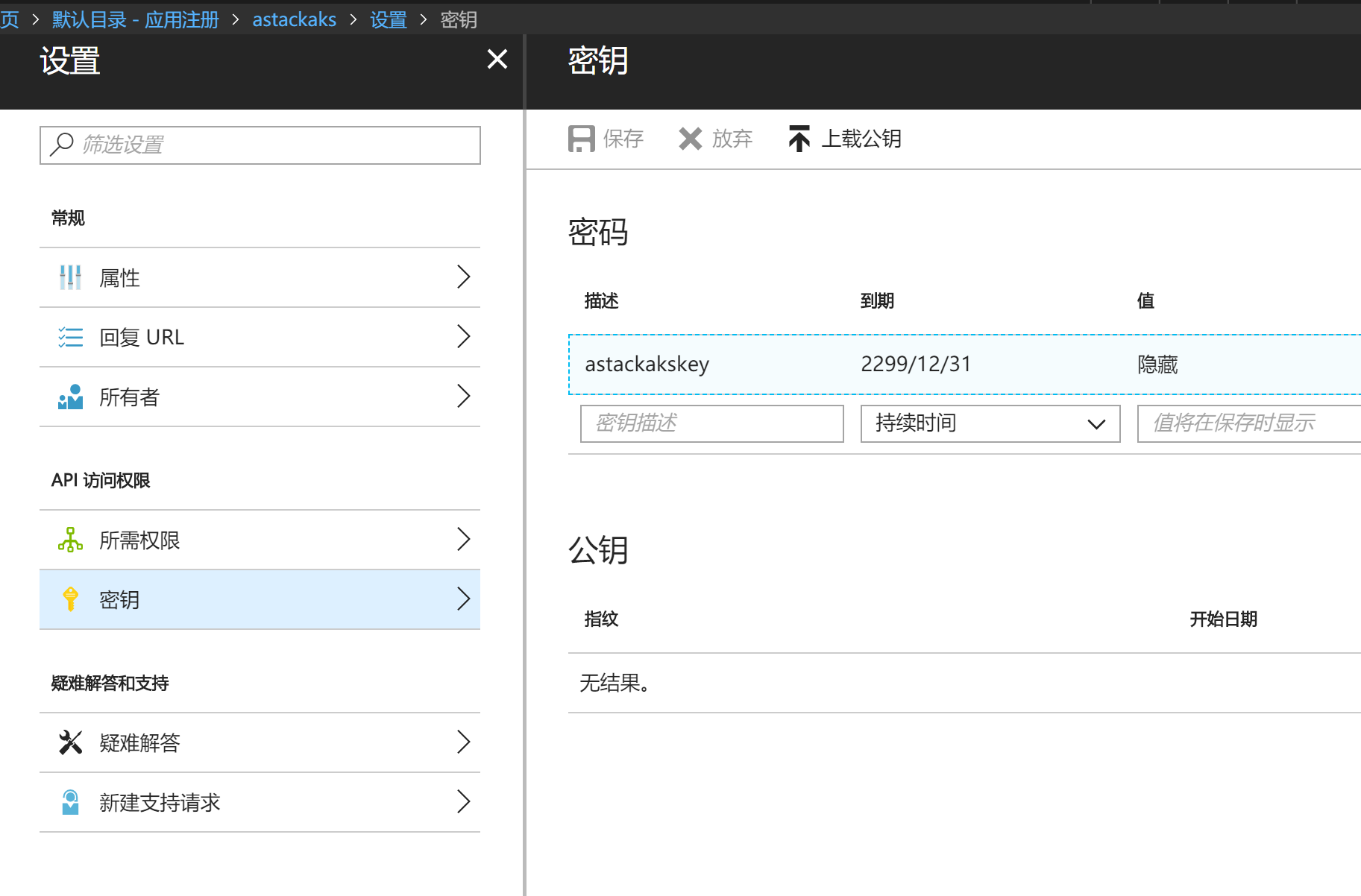
* 用Azure Stack的登录租户ID，登录global Azure
* 进入Azure Active Directory -> App registrations -> New application registration
* 输入应用名称、Application type选择Web app/API、Sign-on URL选择<http://localhost>



* 记录创建的Application ID，这个作为后续要使用的Service Principal client ID。



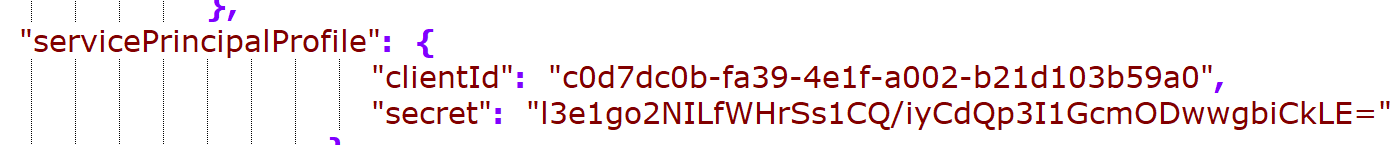
* 点击创建的应用，进入 Settings -> Keys，填写Description、Expire选择Never expires，点击保存，记录创建的Key值。这个作为后续要使用的Service Principal client secret。



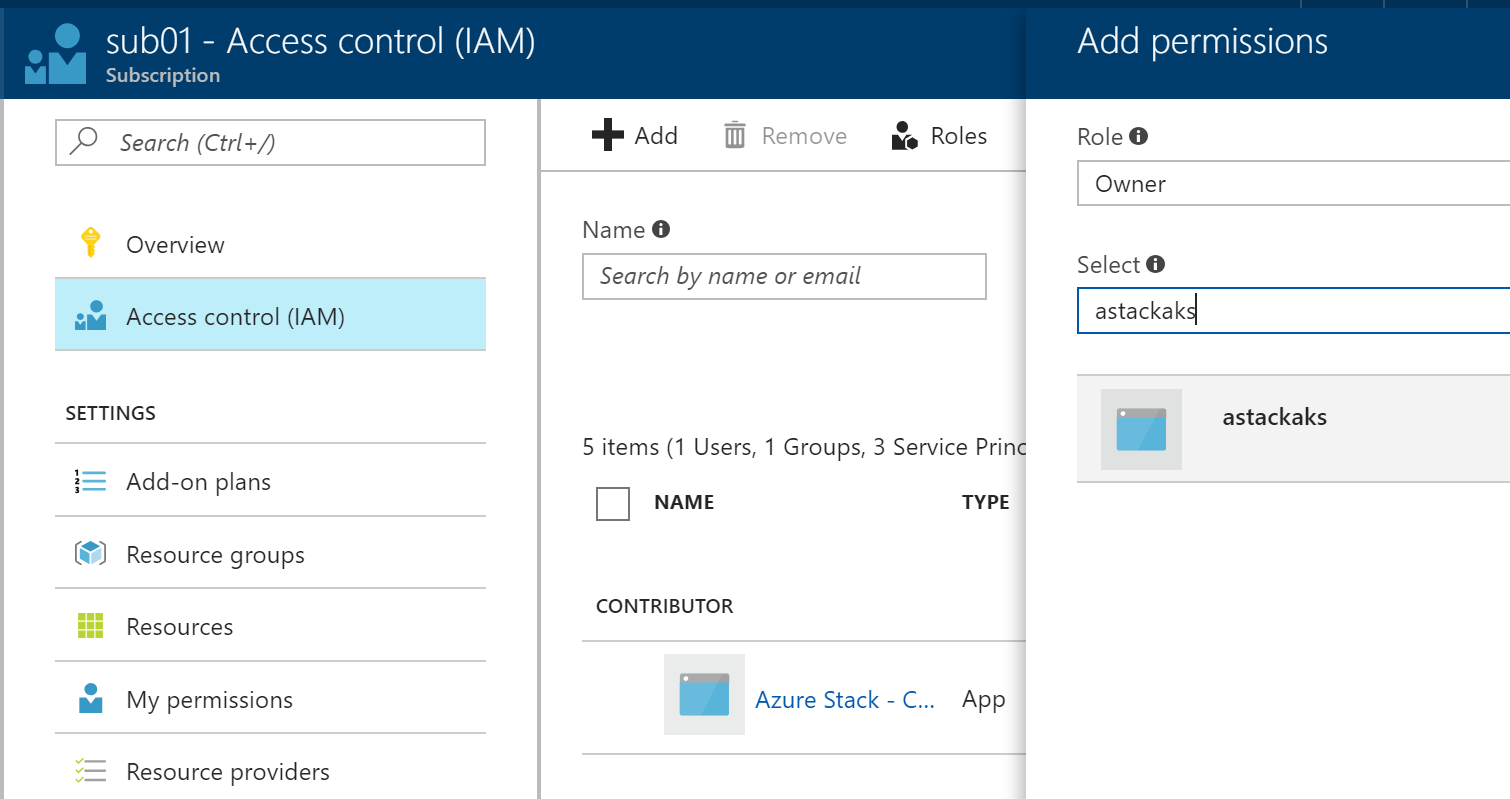
* 打开之前创建的apiModel（azurestack.json）文件，查找到以下部分：



把上面或的client id和key分别填入”clientId”和”secret”



* 给Service Principal授权访问当前Azure Stack的subscription。登入Azure Stack portal，进入Subscription，选择Access control(IAM) -> Add，在Role处选择Owner，在Select下输入应用名进行查找，选择应用并保存



# 创建K8S部署ARM template

* 登入实现准备好的Linux VM，从 <https://github.com/msazurestackworkloads/acs-engine/archive/acs-engine-v0140.zip> 下载acs-engine到本地
* 解压acs-engine-v-140.zip，在examples/azurestack目录下，可以看到acs-engine可执行文件，这个是支持AzureStack的acs-engine版本
* 上传之前生成的apiModel文件到Linux VM
* 运行命令 “./acs-engine generate azurestack.json”（根据当前目录适当调整命令中的路径）
* 在当前目录下，生成\_output目录。在目录下可以找到我们生成的：
  + ARM template：azuredeploy.json
  + Parameter文件：azuredeploy.parameters.json



* 如果Azure Stack位于国内网络环境，为了确保镜像拉取等相关任务的成功，必须要修改ARM template和parameters文件中有关镜像源地址的配置项。具体的做法是：

ARM template：

打开azuredeploy.json，搜索”pod-infra”，并做以下参数修改（共2处）：

--pod-infra-container-image=crproxy.trafficmanager.net:6000/google\_containers/pause-amd64:3.1

Parameters：

打开azuredeploy.parameters.json，做以下修改：

确保将以下参数改为：

"dockerEngineDownloadRepo": {

"value": "https://mirror.kaiyuanshe.cn/docker-engine/apt/repo/"

同时搜索k8s-gcrio.azureedge.net，将其改为“

crproxy.trafficmanager.net:6000/google\_containers”

并将以下参数改为：

"kubernetesTillerSpec": {

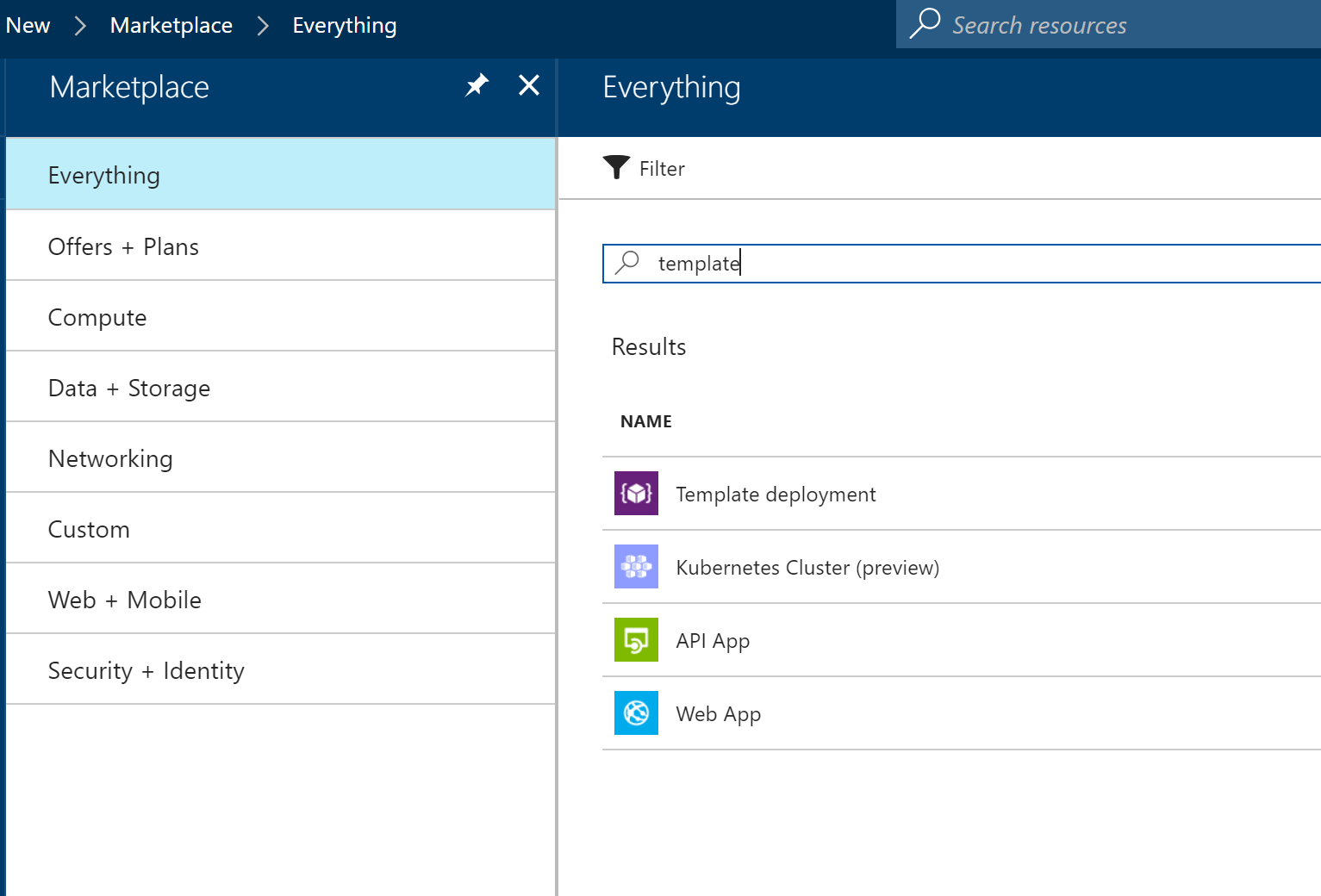
"value": "crproxy.trafficmanager.net:6000/kubernetes-helm/tiller:v2.8.1"

**说明：**

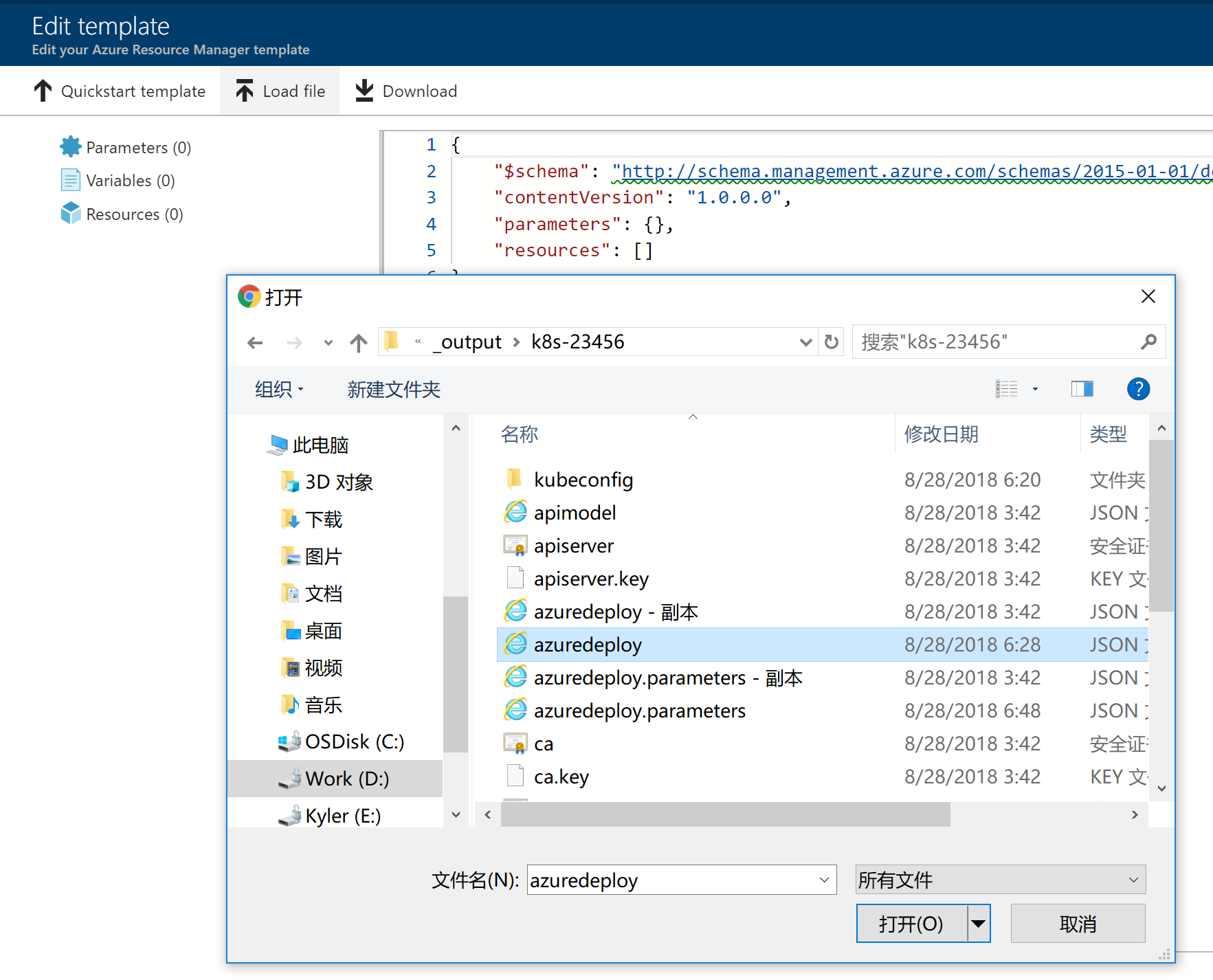
也可以在下一步下载ARM template和parameters到本地后，在到文件中修改镜像源地址的配置项

# 部署template

* 下载上一步创建的ARM template和parameters文件到本地
* 在Azure portal中，搜索”Template deployment”，并选择



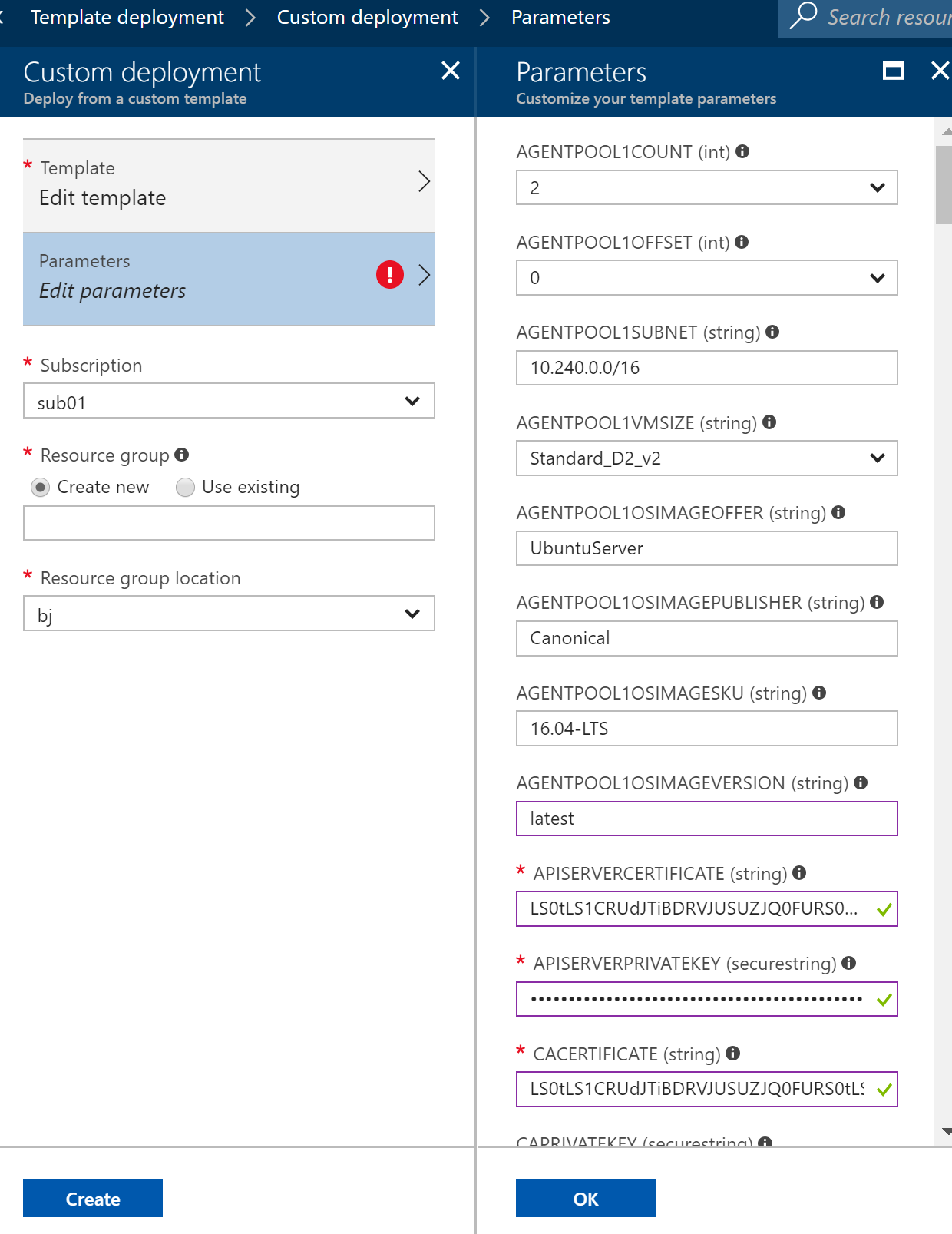
* 在弹出的窗口中，点击第一项编辑template，然后点击上传文件，从本地把修改好参数值的ARM template（azuredeploy.json）上传，保持



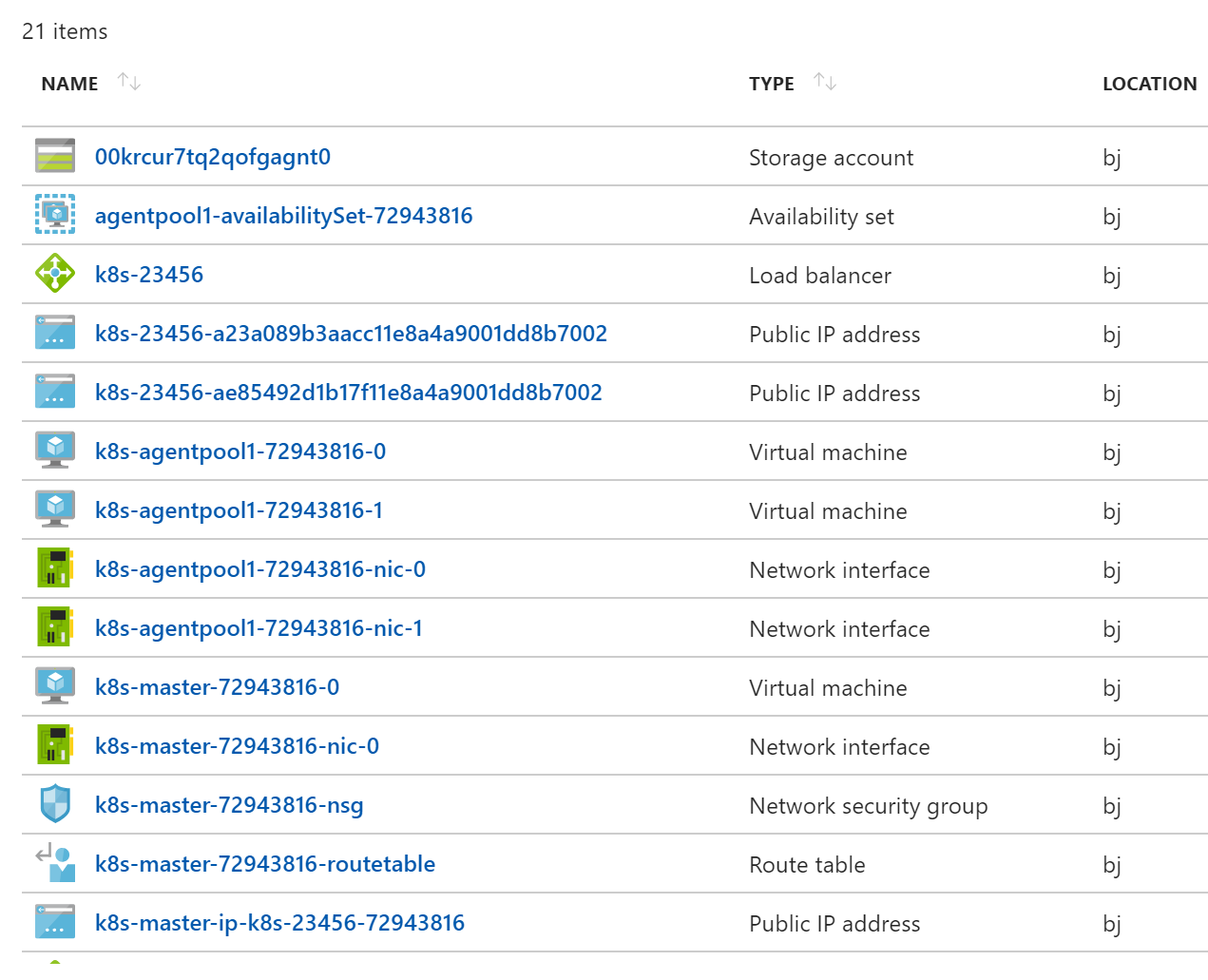
* 点击第二项编辑参数，在右侧的列表中，按照parameters文件中的参数项值，逐个填入列表中。**注意一定要用参数配置文件中的值！**如果列表中某些项已经提供了默认值，请把默认值替换为参数配置文件中相应的值，例如Ubuntu镜像的版本标签，列表中会提供默认值，我们要把默认的版本标签替换成配置文件中的值，即”latest”。

**说明：**

这里手工输入参数是一件比较麻烦和辛苦的工作，一共124项参数，要确保不要输入错误。如果希望用”az group deploy ”命令，需要配置本地的Azure CLI去连接Azure Stack，可以参考link：<https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-version-profiles-azurecli2>）



* 点击OK完成参数列表输入
* 保持当前subscription，输入要安装K8S到的资源组，保持默认的location
* 点击创建按钮进行模板部署
* 进入所选的资源组，能看到K8S集群所需的资源在被逐渐创建出来

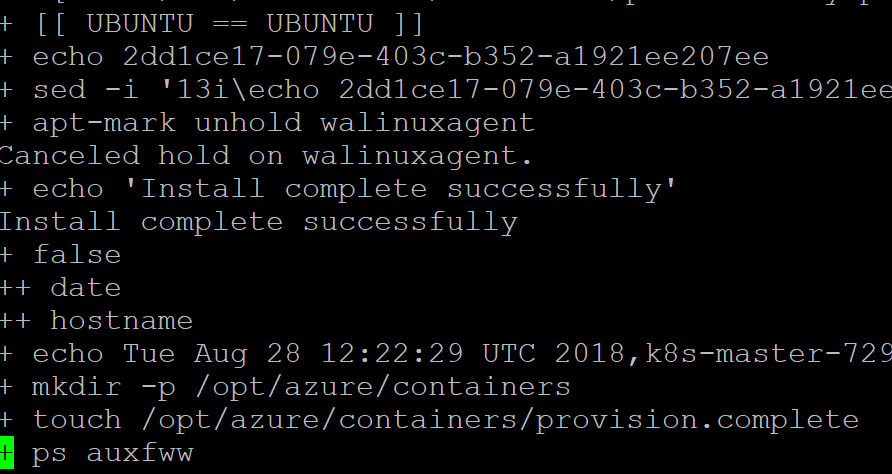


* 进入Deployment，能看见当前deployment的进度和状态

## 确认部署成功

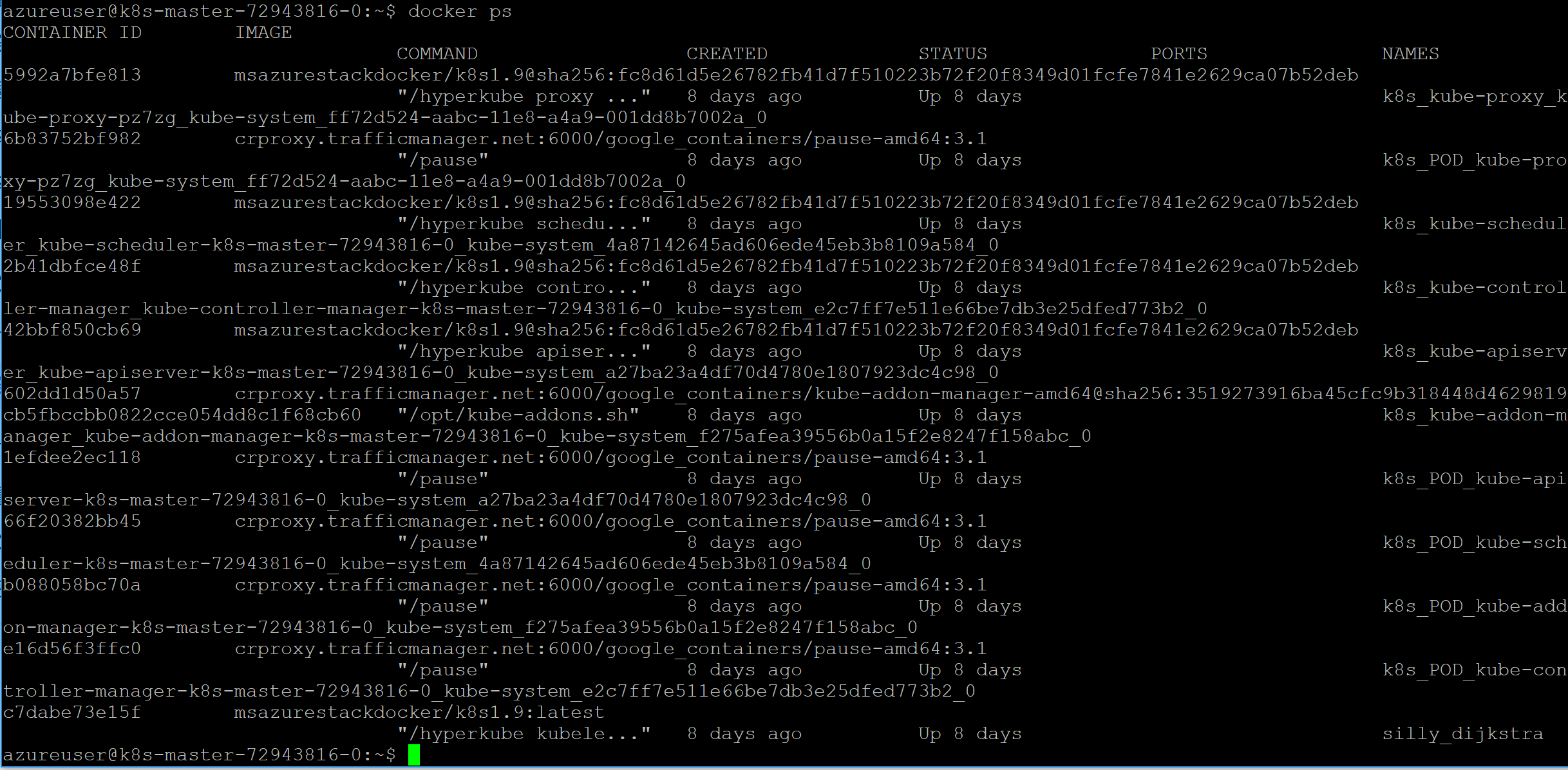
* 部署顺利的话，大约在10~20分钟后，整个部署过程能够结束，且在deployment中看到的所有任务项状态都是OK



* SSH登录master节点，在master节点的 /var/log/azure/cluster-provision.log的最后附近，能看到”Installcomplete successfully”字样

**Master内部部署过程说明：**

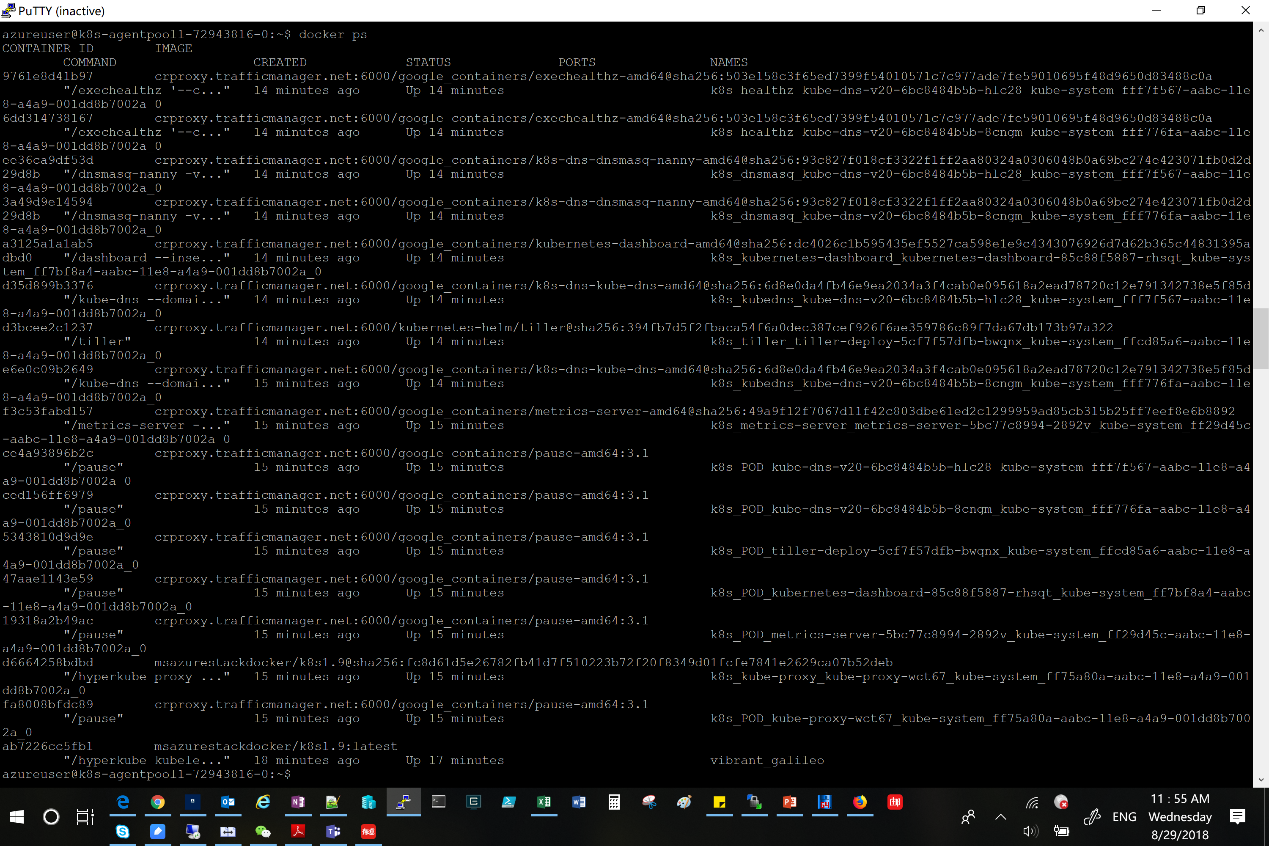
* 安装容器引擎
* 拉取master节点上K8S服务的容器镜像
* 配置集群，启动master节点上的K8S服务，例如api-server, scheduler, controller manager, proxy, add on manager等。在master节点上运行命令”docker ps”，应能看到如下容器运行：



* 安装etcd
* 等待集群中所有slave节点完成安装配置，在指定时间内如果完成启动的节点数量少于预定值，会导致超时，部署失败
* 以上任何一步出错，都可能会导致整个部署过程失败。全部任务成功后，才返回完整部署成功。

**Slave内部部署过程说明：**

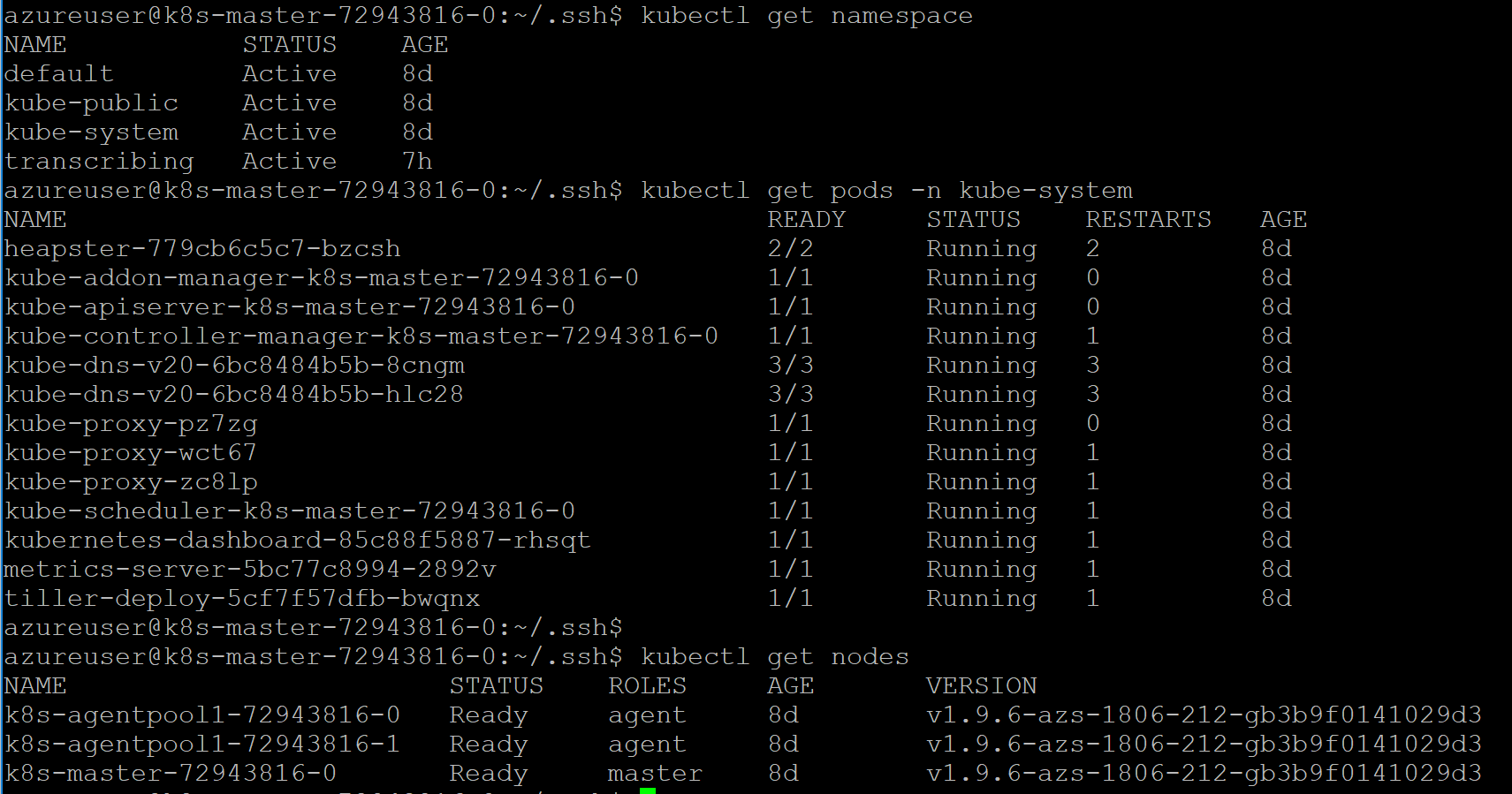
* 安装容器引擎
* 拉取slave节点上K8S服务的容器镜像
* 启动slave节点上的K8S服务，例如dns，proxy，healthz，metrics等。。在slave节点上运行命令”docker ps”，应能看到如下容器运行：



* 看到模板部署成功结束后，在master节点上运行命令”kubectl cluster-info”，能看到主要的部署端点正常运行



* 运行几个命令试验系统是否正常，例如：



## 错误排查

如果遇到问题（例如下面图中的报错），给出的错误信息是conflict，错误码是3，这些信息还不足够定位问题的原因，往往都需要进入到节点上去查找线索。



可以尝试以下方式查找错误线索：

* Master和slave节点上的日志： /var/log/azure/cluster-porvision.log
* 在master节点上运行 “kubectl cluster-info”，查看endpoint是否正常工作。举个例子说明如何排错：通常如果集群部署失败，用kubectl cluster-info返回的信息不正确，或者命令返回连接443端口超时。用命令netstat检查443端口是否处于被监听状态，如果没有443端口被监听，说明api-server没有运行，就需要去查相关的镜像和容器进程是否存在，如果不存在就检查是否镜像源地址不对；如果已经存在，试试重启api-server对应的容器进程等
* 另外，Master和slave接地的部署任务中，某些任务设定有固定的超时时间值，例如容器引擎安装、master节点等待所有slave节点就绪等。在多数情况下，预设定的超时时间够长，很少有在预定时间内完成不了某个任务的情形，但偶尔仍然存在网络访问延迟等问题，导致拉取镜像时间过长而在预定时间内不能完成服务启动的问题。如果怀疑这种可能性，把资源组删除，然后重新用模板部署一遍试试。

## 登录slave节点

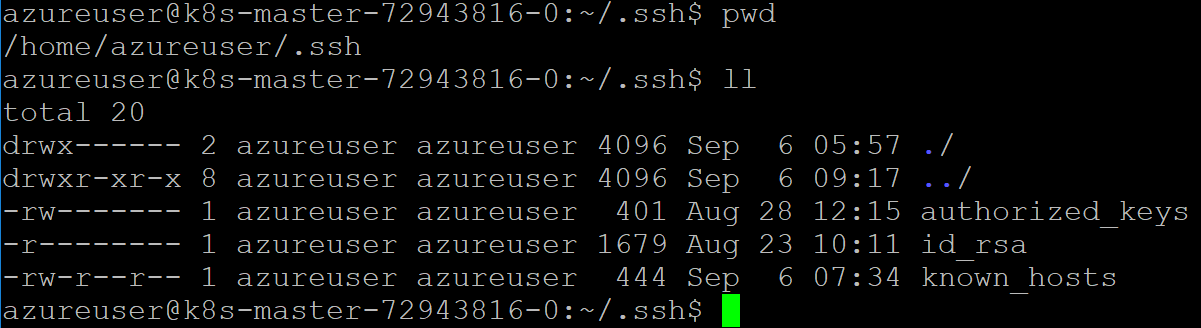
Master节点上提供了很多信息，但有时可能也需要在slave节点上查看日志，运行诊断命令等，这时候就需要SSH登录到slave节点。

在资源组中，我们可以看到master节点有一个public IP，但是还会发现这个public IP实际是分配到master节点前面的LB之上，同时这个LB会设置对端口22/80/443的转发，因此透过master LB的public IP，我们可以访问到master节点。

Slave节点的情况和master节点类似，在portal中查看到的public IP是分配在所有slave节点前面的LB上。和master LB不同的是，slave节点前面的LB除了80端口以外，默认没有配置其它转发端口，因此，虽然看到slave节点有public IP，但无法通过这个public IP访问到slave节点。

建议通过先SSH登录到master节点，然后再从master节点SSH到slave节点各自在vnet中的私有IP。为了从master节点跳转，我们需要：

* 把事先准备好的SSH密钥对里面的公钥上传到master节点的 /home/azureuser/.ssh/目录下（azureuser是用ARM template部署K8S集群时给出的linux管理员用户名，我们也是用这个用户名创建的密钥对，用这个用户名登录的master节点）
* 把公钥文件 id\_rsa的属性设置成 u+r，并为了安全规定，去掉其它所有的属性



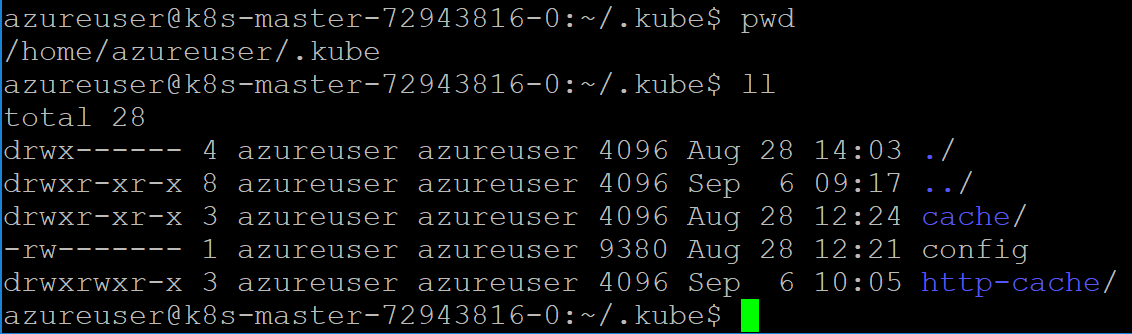
从master节点的命令行运行 ssh <slave\_node\_ip>，即可登录slave节点

# 配置本地连接

Master节点上安装了kubectl工具，但为了方便，我们需要从本地环境直接连接K8S集群

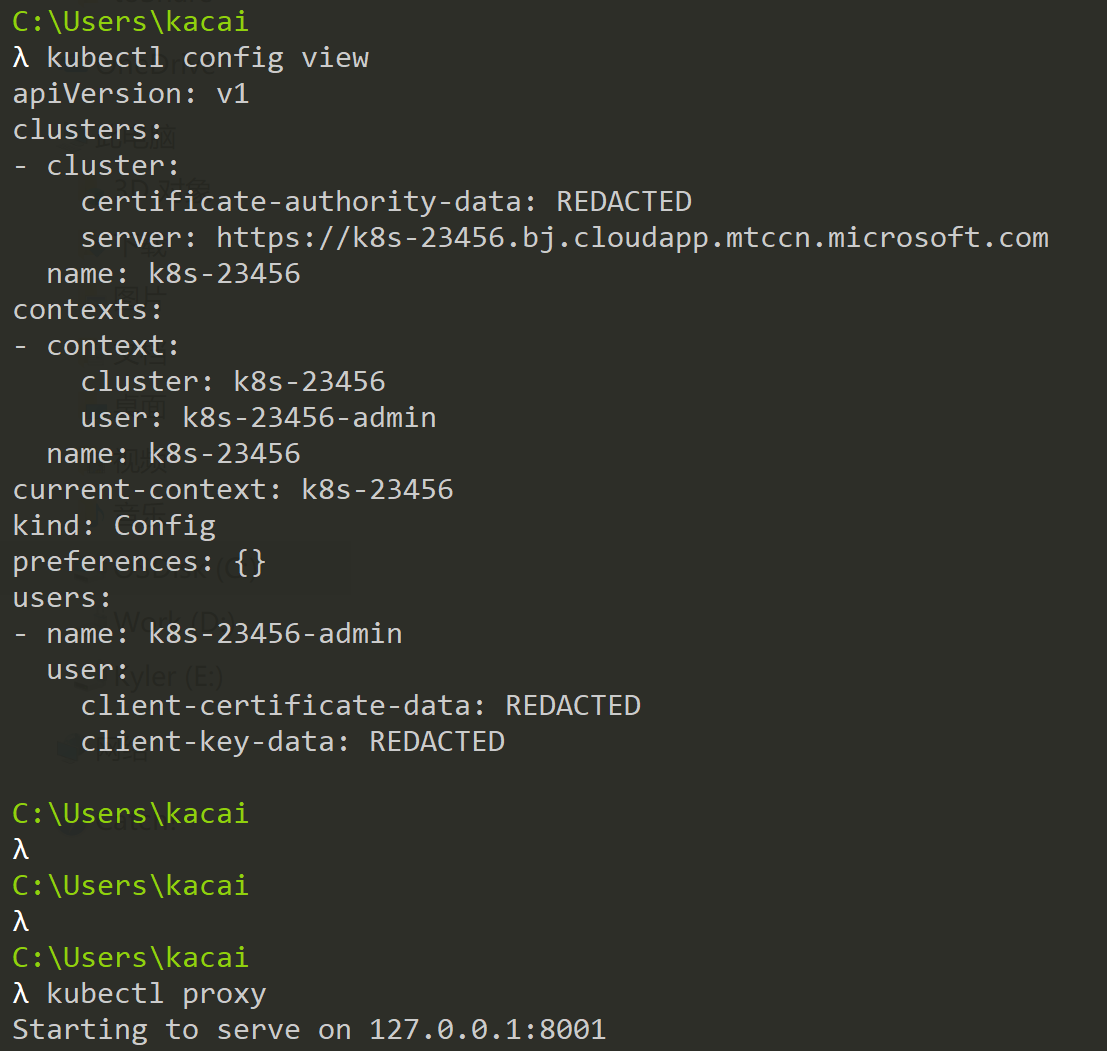
如果希望从本地运行kubectl连接K8S集群，步骤是：

* 确保本地安装了kubectl工具（如还没有安装，请参考link：<https://kubernetes.io/docs/tasks/tools/install-kubectl/>，或者如果已经有Azure CLI，可以运行命令”az aks install-cli”）
* 从master节点的 /home/azureuser/.kube 目录下载config文件到本地，放置在当前用户profile的 .kube目录下，例如” C:\Users\kacai\.kube”



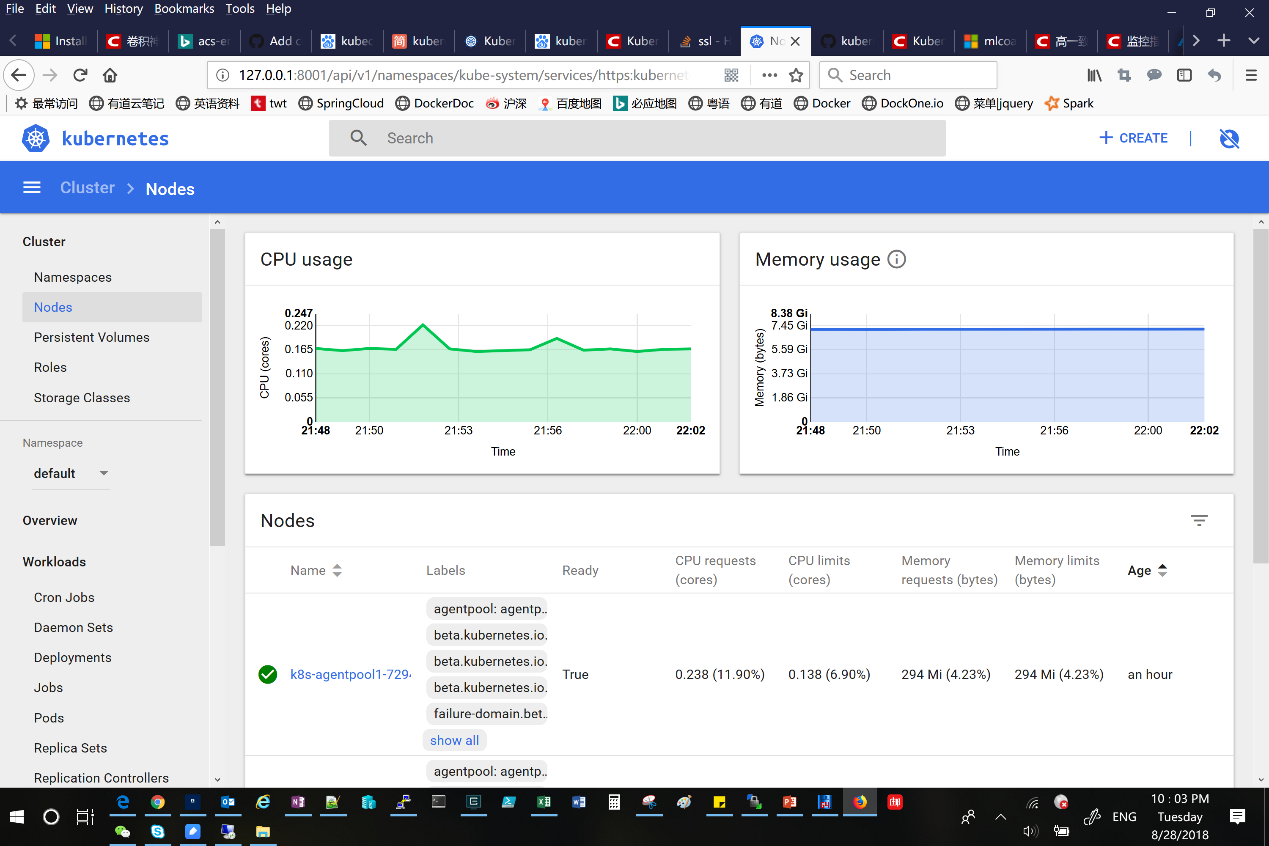
这样就可以在本地运行kubectl工具连接安装好的K8S集群了

* 运行”kubectl config view”查看K8S的配置，验证本地kubectl连接K8S正常



除了命令行，如果需要访问K8S的dashboard，临时但是最简单做法是：

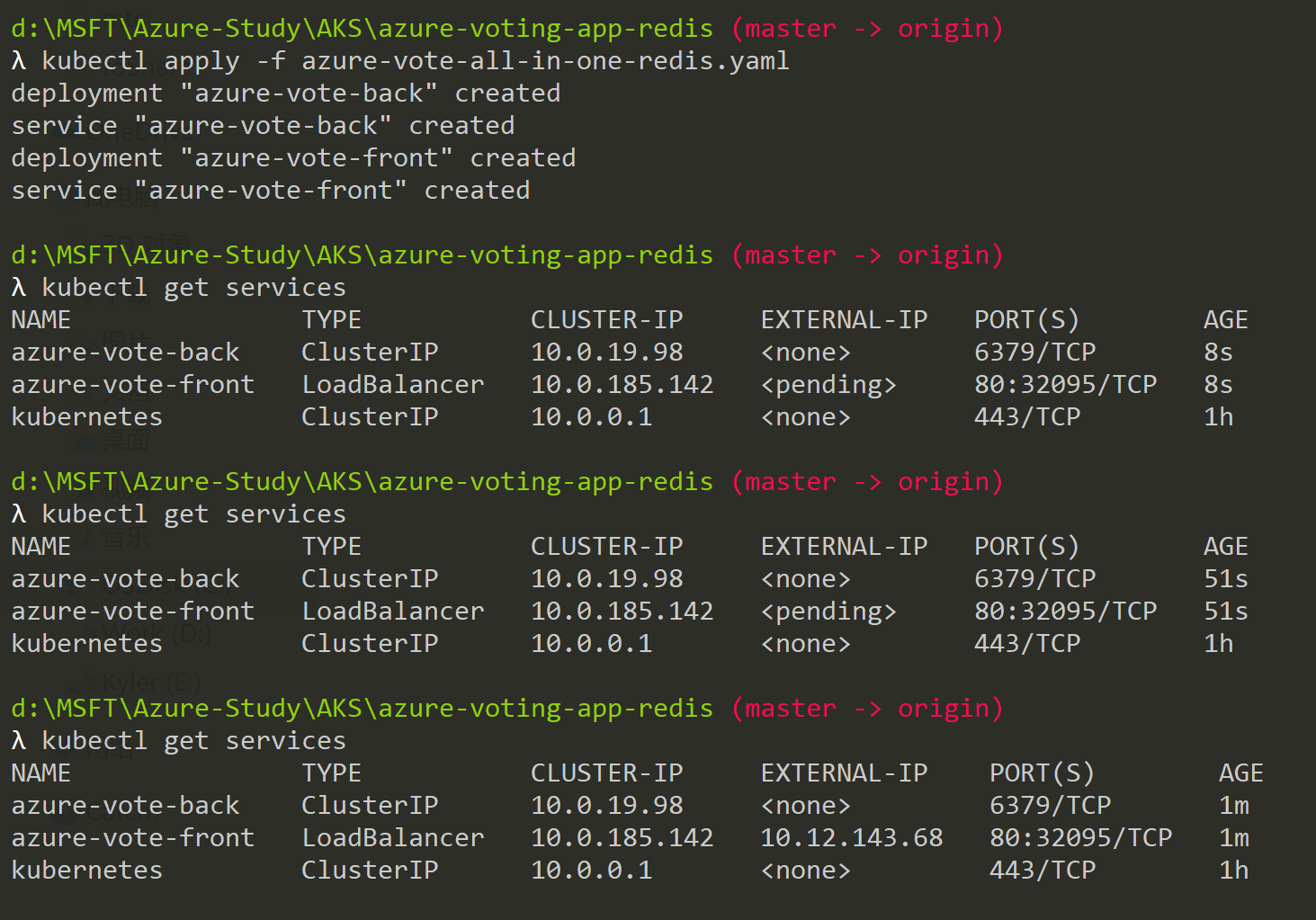
* 在本地运行”kubectl proxy”，在本地启动对api-server的代理，命令返回提示proxy监听到本地的8001端口
* 打开浏览器，输入地址：<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>
* 如果需要验证，选择跳过，就可以看到K8S的dashboard了



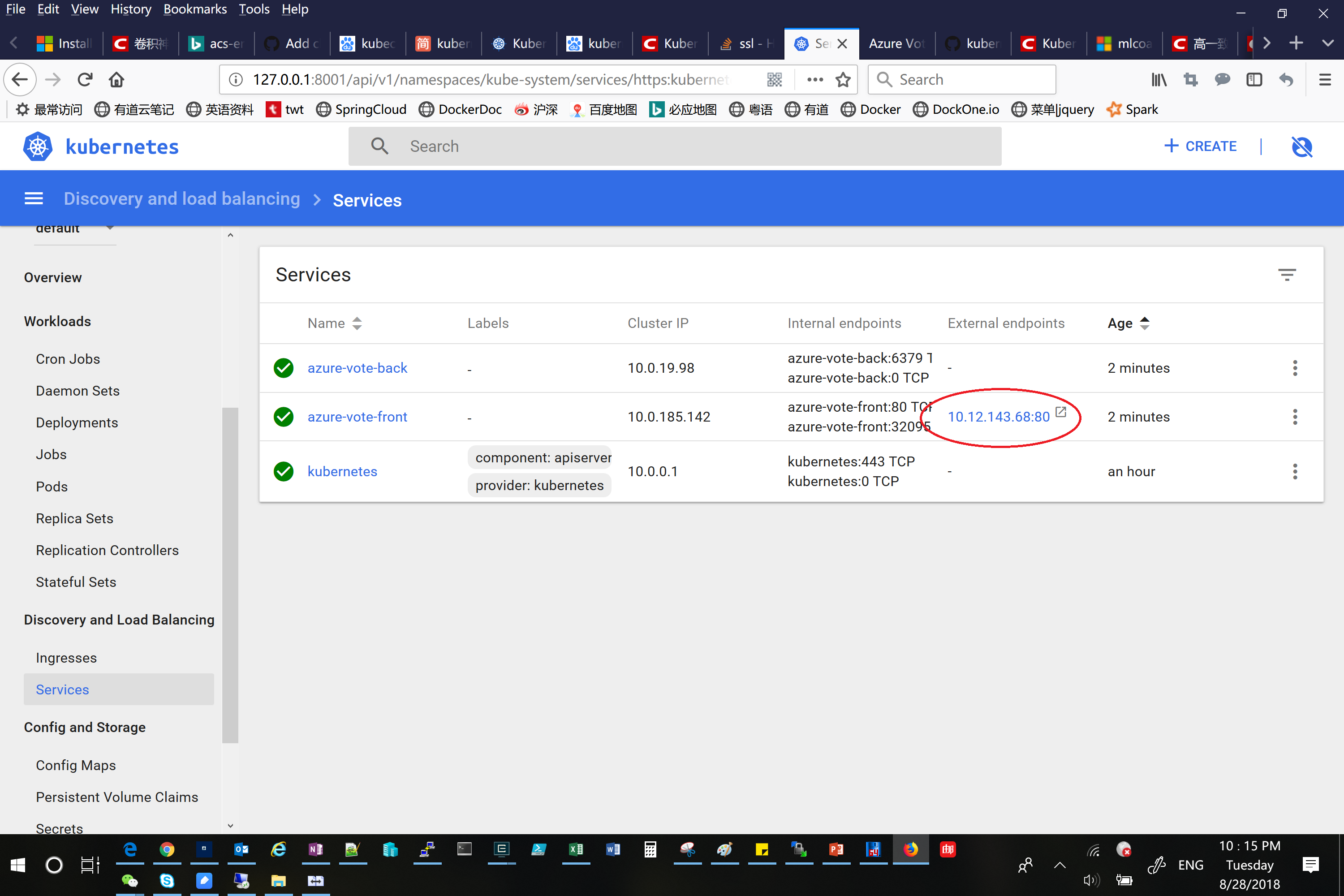
# 部署应用

我们来部署一个应用，检验在Azure Stack上部署的K8S cluster是否工作正常

* 在从<https://github.com/kylercai/K8SOnAzureStack.git>克隆的文件中，包含一个azure-vote-all-in-one-redis.yaml。这是个投票应用，包含前后端两个服务，以及服务各自对应的deployment对象的manifest文件。我们在做AKS的练习时候，已经用过它，我们在这里可以直接借过来用用。
* 执行命令”kubectl apply -f azure-vote-all-in-one-redis.yaml”，把这个投票应用部署到K8S集群
* 执行命令”kubectl get services”，一开始azure-vote-front的EXTERNAL-IP字段是pending状态，这是正在为azure-vote-front准备LoadBalancer。一段时间后，当发现azure-vote-front的EXTERNAL-IP字段变为IP地址后，就可以通过这个IP地址和给出的端口访问服务了



或者我们可以在dashboard中，进入Services面板，看见azure-vote-front服务已经就绪，点击External endpoints中对应的地址，即可访问azure-vote-front的前端页面：



看到azure-vote-front的前端页面了，说明我们在Azure Stack上部署的K8S工作正常。



至此，我们在Azure Stack上部署K8S集群的工作就成功了。

# 参考信息

ACS on AzureStack

<https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-template-kubernetes-deploy>

Add a Kubernetes Cluster to the Azure Stack Marketplace

<https://docs.microsoft.com/en-us/azure/azure-stack/azure-stack-solution-template-kubernetes-cluster-add>

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal#check-azure-active-directory-permissions>

微信公众号文章：

在Azure Stack上部署Kubernetes

<https://mp.weixin.qq.com/s?__biz=MzA3NTM1MzE4Nw==&mid=2649617738&idx=1&sn=72e614800b05450390348b290a5b13e7&chksm=87688846b01f015006aeeaa32850fd7d2c8660599e026738b0845e4c908bb0ec98b135408ae9&mpshare=1&scene=1&srcid=0801r6UviPLPMgo9oEcOpUMD#rd>

有关中国Azure Stack部署K8S的疑难解答

<https://mp.weixin.qq.com/s?__biz=MzA3NTM1MzE4Nw==&mid=2649617746&idx=1&sn=8d110646120244bbab6b55e252fc744e&chksm=8768885eb01f014824a130cb4148cd9c8648e20d155549a1d09df7d88b3dceeec64821ee09dd#rd>

AzureStack K8S templates：

<https://github.com/radhikagupta5/AzureStack-QuickStart-Templates/tree/radhikgu-acs/101-acsengine-kubernetes-1803>

<https://github.com/radhikagupta5/AzureStack-QuickStart-Templates/tree/radhikgu-acs/101-acsengine-kubernetes-1804>

<https://github.com/ahpeng/K8SOnAzureStack>

步骤参考:

<https://github.com/Azure/acs-engine/tree/master/docs>

\* <https://github.com/radhikagupta5/AzureStack-QuickStart-Templates/tree/radhikgu-acs/101-acsengine-kubernetes-1804>

\* <https://mp.weixin.qq.com/s?__biz=MzA3NTM1MzE4Nw==&mid=2649617738&idx=1&sn=72e614800b05450390348b290a5b13e7&chksm=87688846b01f015006aeeaa32850fd7d2c8660599e026738b0845e4c908bb0ec98b135408ae9&mpshare=1&scene=1&srcid=0801r6UviPLPMgo9oEcOpUMD>

\* <https://github.com/radhikagupta5/AzureStack-QuickStart-Templates/tree/radhikgu-acs/101-acsengine-kubernetes-1803>

\* <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-template-kubernetes-deploy>

错误参考：

<https://github.com/Azure/acs-engine/issues/2591>

<https://github.com/Azure/acs-engine/issues/1806>