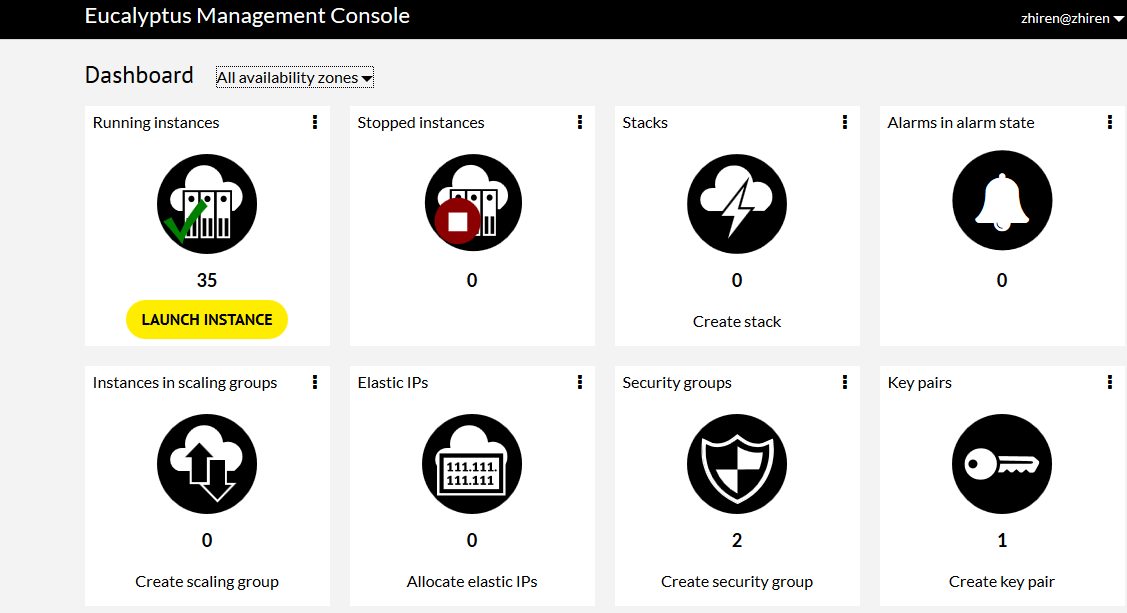
## CI环境创建

### Cloud instance创建

#### 打开<http://gerrit.nsn-net.net/gitweb?p=TDDLTE/Tools/CIScripts.git;a=blob;f=CLOUD/readme.txt;hb=refs/heads/master> ， 选择“Cloud information”段中定义的任何一个cloud，打开URL登入，画面如下



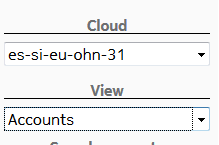
#### (Optional)点击create security group去创建要暴露的的端口

可以参考已经存在的security group

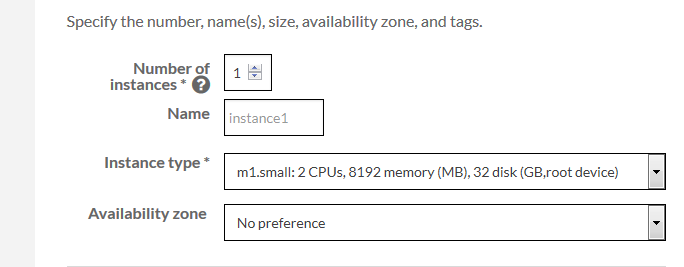
#### 点击LAUNCH INSTANCE去创建instance

选择image，也就是OS

选择资源， 注意选择资源不能超过cloud上剩余资源限制，link查看<https://status.eecloud.nsn-net.net/cloud/> , 输入你的cloud名字，view选择Accounts,如下



选择资源如下：

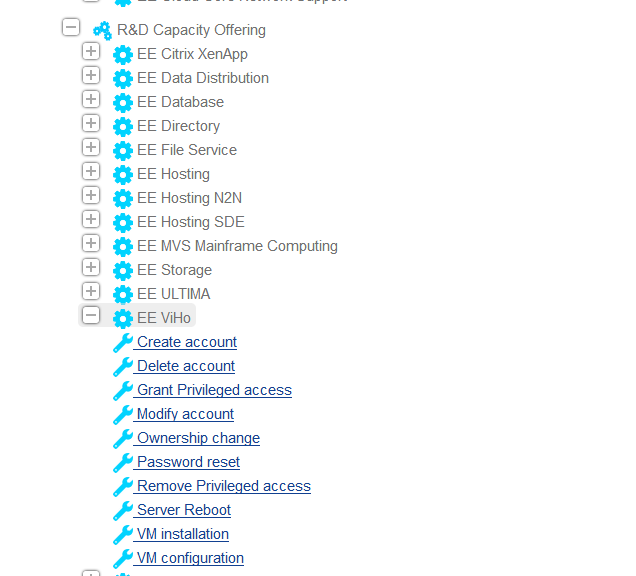


点击lanch new instance就OK了。

### Instance的安装

## Kubernetes

### Kubernet下给cluster范围内的所有pod分配cluster范围内的私有ip， 这个私有ip在cluster范围内是唯一的， 因此不同node下的的container能否互相访问的。



### Node， pod， container

Pod运行在node上，当宿主node fail后，该pod可以迁移到其他的node上。

一般，一个pod下一个container， 也可以是一个pod下多个container，通常这些container有着紧密的关系。例如一个container提供share文件访问和更新，一个提供network访问。

### Controller和pod

通过controller去创建pod， controller使用deployment去创建pod。Deployment内容如下

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2

kind: Deployment

metadata:

name: nginx-deployment //该deployment的名字

labels:

app: nginx //给本deployment打上label

spec:

replicas: 3 //本deployment创建3个replicated pods

selector: //创建pods后，该deployment怎样去找到这些pods去管理

matchLabels:

app: nginx //带有lable为app=nginx的pod会被选中去管理。

template: //这些pod运行在哪样的container上

metadata:

labels:

app: nginx //似乎是查找pod重复

spec:

containers:

- name: nginx //运行在名为nginx的container上？不太可能三个container名字一样吧？

image: nginx:1.7.9 //该container运行这个版本的image

ports:

- containerPort: 80 //该container打开80端口用于发送和接收traffic

保存为yaml文件，run by kubectl.

kubectl create -f https://raw.githubusercontent.com/kubernetes/website/master/docs/concepts/workloads/controllers/nginx-deployment.yaml

可以通过kubectl get deployments来show出来有哪些deployment被创建。

## Docker

### 解决docker EE在CentOS上启动错误

level=fatal msg="Error starting daemon: error initializing graphdriver: loo

pback mounting failed"

for i in {0..6}

do

mknod -m0660 /dev/loop$i b 7 $i

done

### 修改 docker daemon的配置参数

如： 修改docker使用指定的disk和硬盘驱动：

修改/etc/docker/daemon.jason为

{

"graph": "/mnt/docker-data",

"storage-driver": "overlay"

}

如给docker daemon加上proxy

<https://docs.docker.com/engine/admin/systemd/#httphttps-proxy>

### <https://docs.docker.com/engine/userguide/storagedriver/selectadriver/#supported-backing-filesystems>

此表似乎不对

| **Storage driver** | **Supported backing filesystems** |
| --- | --- |
| overlay, overlay2 | ext4, xfs |
| aufs | ext4, xfs |
| devicemapper | direct-lvm |
| btrfs | btrfs |
| zfs | zfs |

我的redhat7.4环境：

DOCKER\_STORAGE\_OPTIONS="--storage-driver devicemapper --storage-opt dm.fs=xfs --storage-opt dm.thinpooldev=/dev/mapper/vg00-docker--pool --storage-opt dm.use\_deferred\_removal=true --storage-opt dm.use\_deferred\_deletion=true

### 我们自己安装docker的方法

我们在redhat上使用下面方法

tee /etc/yum.repos.d/docker.repo <<-'EOF'

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF

### install docker

yum install -y docker-engine

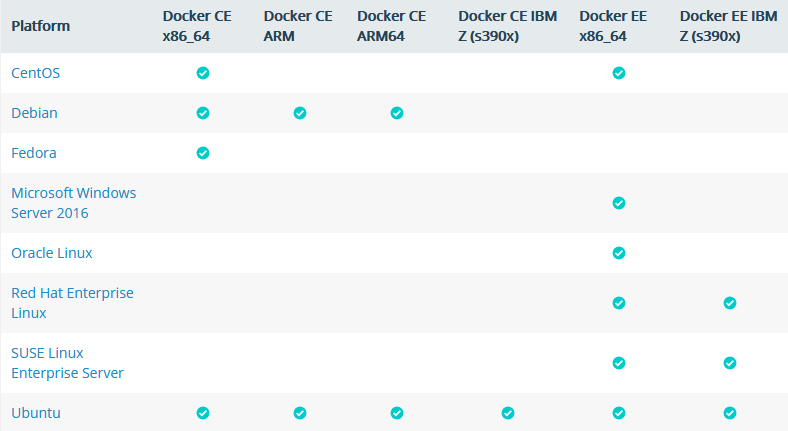
####start docker

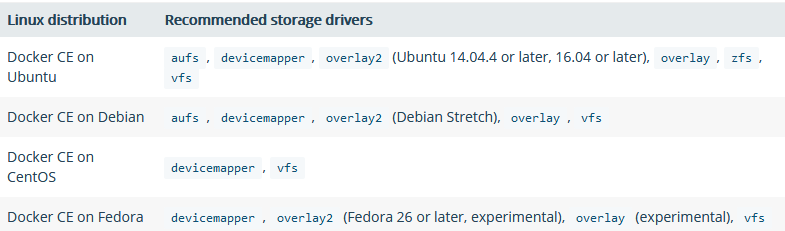
systemctl enable docker

systemctl start docker

与官网上的restriction不一致， 而且我们现在的redhat上跑centos上的CE 版本的docker。而且这个docker版本居然自动使用overlay的storage driver，而且能启动成功。

Tips:我开始在hzcisv01.china.nsn-net.net装EE 版本docker， ee docker使用storage-drive是正确的devicemapper， 但是后来我follow上面的指导， 发现CE的docker在redhat上启动失败， 原因居然是上次EE的docker遗留文件夹devicemapper于/var/lib/docker下， 造成了CE docker依然使用device mapper启动， 却启动失败， 把这个遗留devicemapper给删掉，居然将错就错得启动OK了





### 设置docker registry的命令

INSECURE\_REGISTRY='--insecure-registry rcp-docker.eecloud.dynamic.nsn-net.net:5000' 似乎是此命令

### 原来RCP的docker设置

是于CentOS上安装EE 1.10的docker版本， 而且storage driver使用的正确的device-maaper， 但是真实用于存储data和metadata的依然是loop device而不是真实的硬盘。但是似乎loop device做了一些配置。

[hongjunyang@euca-172-31-9-44 hongjyan]$ sudo docker info

Containers: 30

Running: 9

Paused: 0

Stopped: 21

Images: 20

Server Version: 1.10.3

Storage Driver: devicemapper

Pool Name: docker-253:33-6553601-pool

Pool Blocksize: 65.54 kB

Base Device Size: 16.11 GB

Backing Filesystem: xfs

Data file: /dev/loop0

Metadata file: /dev/loop1

Data Space Used: 55.79 GB

Data Space Total: 107.4 GB

Data Space Available: 51.58 GB

Metadata Space Used: 37.92 MB

Metadata Space Total: 2.147 GB

Metadata Space Available: 2.11 GB

Udev Sync Supported: true

Deferred Removal Enabled: false

Deferred Deletion Enabled: false

Deferred Deleted Device Count: 0

Data loop file: /docker/devicemapper/devicemapper/data

WARNING: Usage of loopback devices is strongly discouraged for production use. Either use `--storage-opt dm.thinpooldev` or use `--storage-opt dm.no\_warn\_on\_loop\_devices=true` to suppress this warning.

Metadata loop file: /docker/devicemapper/devicemapper/metadata

Library Version: 1.02.107-RHEL7 (2016-06-09)

Execution Driver: native-0.2

Logging Driver: json-file

Plugins:

Volume: local

Network: bridge null host

Kernel Version: 3.10.0-327.36.3.el7.x86\_64

Operating System: CentOS Linux 7 (Core)

OSType: linux

Architecture: x86\_64

Number of Docker Hooks: 2

CPUs: 2

Total Memory: 7.797 GiB

Name: euca-172-31-9-44

ID: VHQL:LCPG:N6EU:ZBCF:PVH3:3TWG:PCYP:7KCG:UPBA:YDXJ:SZFO:TA4G

WARNING: bridge-nf-call-iptables is disabled

WARNING: bridge-nf-call-ip6tables is disabled

Registries: docker.io (secure)

[hongjunyang@euca-172-31-9-44 hongjyan]$ cat /etc/redhat-release

CentOS Linux release 7.2.1511 (Core)

[hongjunyang@euca-172-31-9-44 hongjyan]$ cat /etc/sysconfig/docker

# /etc/sysconfig/docker

# Modify these options if you want to change the way the docker daemon runs

#OPTIONS='--selinux-enabled --log-driver=journald'

#DOCKER\_CERT\_PATH=/etc/docker

OPTIONS=' -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock --storage-opt dm.basesize=15G -g /docker'

INSECURE\_REGISTRY='--insecure-registry rcp-docker.eecloud.dynamic.nsn-net.net:5000'

DOCKER\_CERT\_PATH=/etc/docker

### Redhat7下EE Docker下删除/var/lib/docker目录后，docker不能启动

会出现下面错误:

Unable to take ownership of thin-pool (rootvg-docker--pool) that already has used data blocks

原因是docker data里有数据了，但是metadata数据没有。

方法是先删除这个thin-pool。

lvremove docker-pool

lvcreate --type thin-pool --size 13000M --name docker-pool vg00

### Docker启动失败Error response from daemon: devmapper: Unknown device e3ad7f74751e51abd16f54884028a9acbecde6a08ec8e159de7d3f0874685330.

[root@hzcisv01 ~]# docker run hello-world

/usr/bin/docker-current: Error response from daemon: devmapper: Unknown device e3ad7f74751e51abd16f54884028a9acbecde6a08ec8e159de7d3f0874685330.

先stop docker， 然后清除/var/lib/docker/下的所有数据，然后重启docker

### 查看具体某容器的配置

/var/lib/docker/containers/019c129d0a9eae13a8fd9fbbd07f326fae0db0c88e825b068ee78721d111353c/

## XML&HTML

### HTML中URL中特殊字符

#： 用来进一步的指定子section。

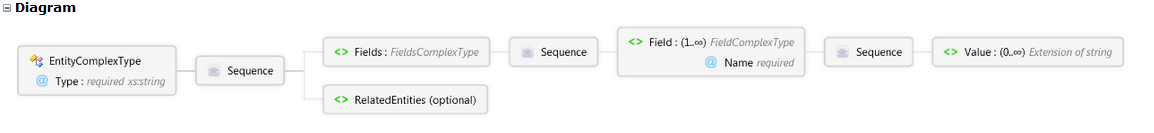
=： 用来给某个name指定value

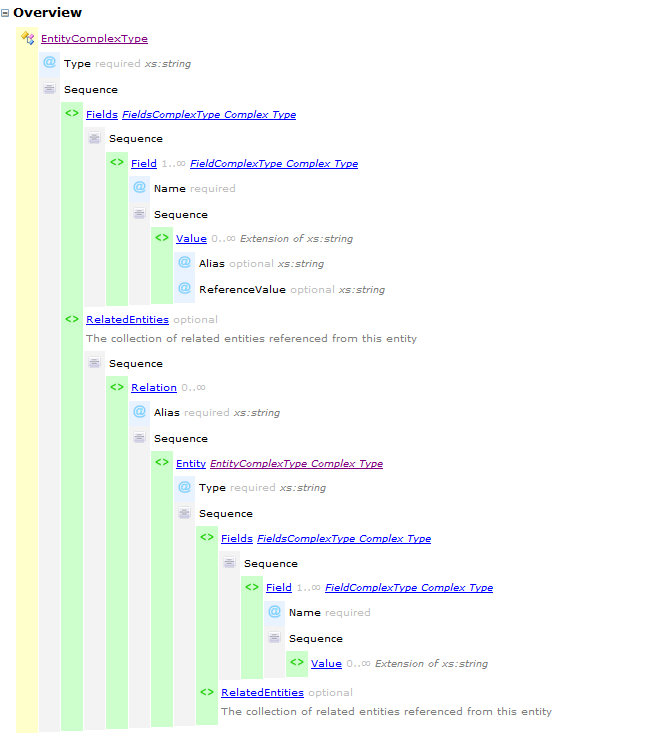
空格不允许出现在URL中

对于以上的特殊字符的literal meaning， 需要对其进行编码， 如SPACE被encode为’+’或者“%20”

### XML schema和JAVA结构的对应

#### JAVA语法





#### Source如下：

<xs:complexType name="EntityComplexType" xmlns:xs="http://www.w3.org/2001/XMLSchema">

//哪里是type？在倒数第二行

<xs:sequence>

<xs:element name="Fields" type="FieldsComplexType" /> //无value的Field？似乎是的

<xs:element name="RelatedEntities" minOccurs="0"> //并行的relatedentities， optional

<xs:annotation>

<xs:documentation>

The collection of related entities referenced from this entity

</xs:documentation>

</xs:annotation>

<xs:complexType> //找不到对应的定义，似乎sequence是封装在complexType下的

<xs:sequence>

<xs:element name="Relation" minOccurs="0" maxOccurs="unbounded"> //releation， 无Alias

<xs:complexType> //找不到对应定义，难道这个对应整个Alias

<xs:sequence>

<xs:element name="Entity" type="EntityComplexType" /> //对应Entity

</xs:sequence>

<xs:attribute name="Alias" type="xs:string" use="required" />

</xs:complexType>

</xs:element> //sequence下Relation定义结束

</xs:sequence> //RelatedEntities下的sequnce结束

</xs:complexType> //似乎RelatedEntities下的sequence是封装在这个complexType下的

</xs:element>//RelatedEntities这个element结束

</xs:sequence> //EntityComplexType下的sequence结束

<xs:attribute name="Type" type="xs:string" use="required" /> //EntityComplexType下的Type

</xs:complexType>

#### 返回结果in JASON

{"entities":

[

{"Fields":

[

{"Name":"estimate-devtime","values":[]},//0..无限号：property名字+s

{"Name":"vc-checkin-time","values":[{"value":"17:12:44"}]},

{"Name":"vc-time","values":[{}]},

{"Name":"base-test-id","values":[]},

{"Name":"storage-path","values":[{}]},

{"Name":"vc-start-audit-action-id","values":[{"value":"3"}]},

{"Name":"configurations-count","values":[{"value":"1"}]},

{"Name":"id","values":[{"value":"1"}]},

{"Name":"vc-comments","values":[{}]},

{"Name":"description","values":[{"value":"\r\n\r\n&gt;test1 desc\r\n\r\n"}]},

{"Name":"name","values":[{"value":"Test1"}]},

{"Name":"has-linkage","values":[{"value":"N"}]},

{"Name":"vc-status","values":[{"value":"Checked\_In"}]},

{"Name":"has-criteria","values":[{"value":"N"}]},

{"Name":"dev-comments","values":[{"value":"\r\n\r\ntest 1 comment\r\n\r\n"}]},

{"Name":"vc-version-number","values":[{"value":"2"}]},

{"Name":"vc-checkin-user-name","values":[{"value":"sa"}]},

{"Name":"creation-time","values":[{"value":"2012-01-01"}]},

{"Name":"template","values":[{}]},

{"Name":"last-modified","values":[{"value":"2012-01-01 17:14:31"}]},

{"Name":"status","values":[{"value":"Design"}]},

{"Name":"has-dependencies","values":[{"value":"0"}]},

{"Name":"attachment","values":[{}]},

{"Name":"steps","values":[{"value":"0"}]},

{"Name":"runtime-data","values":[{}]},

{"Name":"check-out-user-name","values":[{}]},

{"Name":"vc-checkin-date","values":[{"value":"2012-01-01"}]},

{"Name":"subtype-id","values":[{"value":"MANUAL"}]},

{"Name":"vc-end-audit-action-id","values":[{"value":"3"}]},

{"Name":"parent-id","values":[{"value":"1001"}]},

{"Name":"exec-status","values":[{"value":"Passed"}]},

{"Name":"test-ver-stamp","values":[{"value":"13"}]},

{"Name":"vc-checkin-comments","values":[{"value":"t 1 checked in"}]},

{"Name":"vc-date","values":[]},

{"Name":"owner","values":[{"value":"sa"}]},

{"Name":"text-sync","values":[{}]},

{"Name":"step-param","values":[{"value":"0"}]},

{"Name":"timeout","values":[]}

],

"Type":"test"

}

{"Fields":

[

... Repeat for each entity

],

"Type":"test"

}

],

"TotalResults":1

}