CROSSENT
ross the bridge to the future

Kubernetes Tenant Setup & CLI

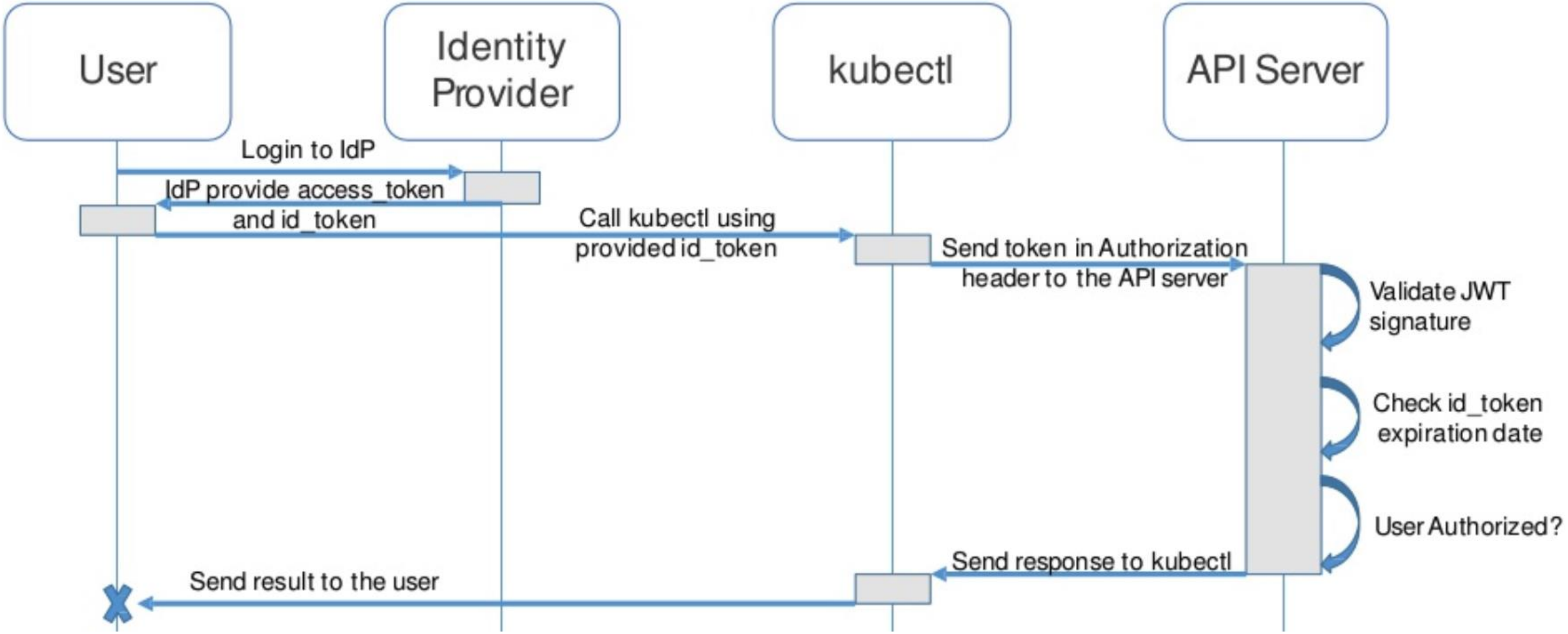
Abhilash S

㈜크로센트
2018. 07

Agenda

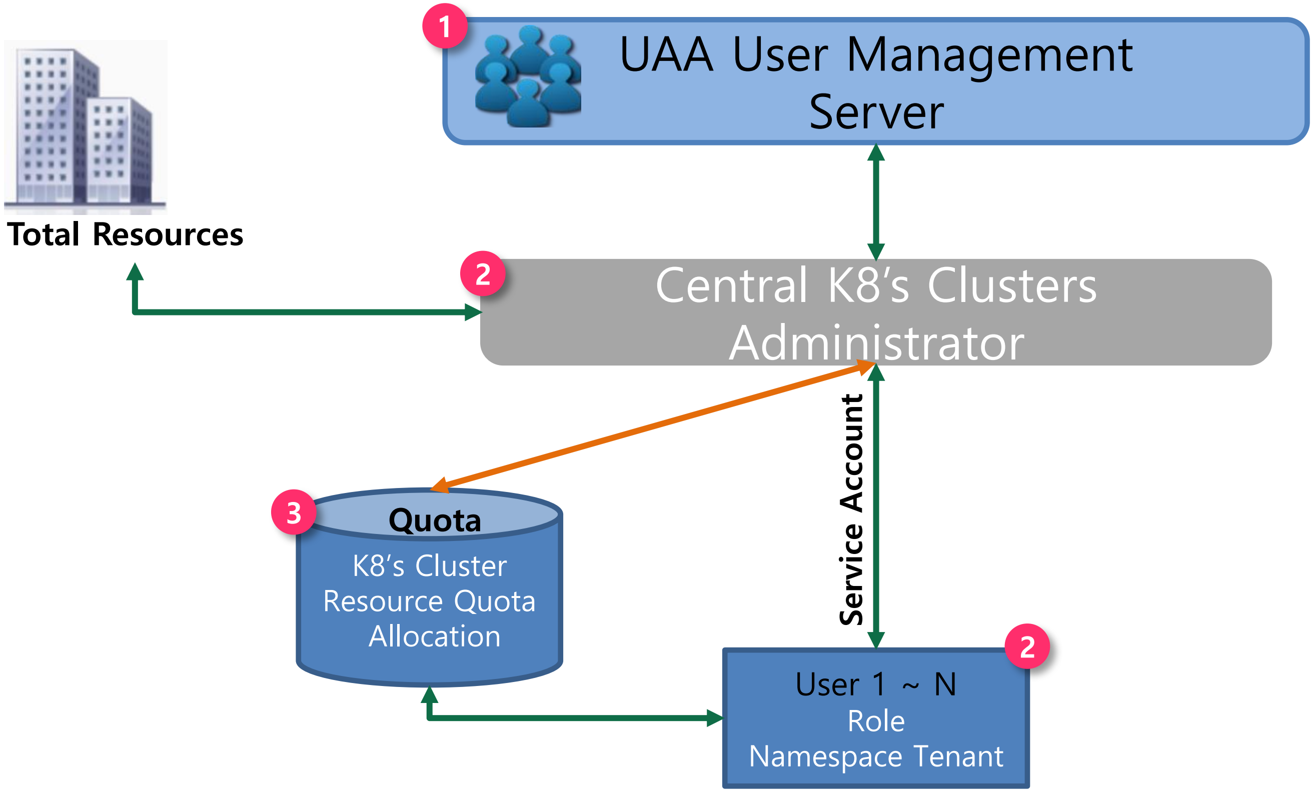
1. Kubo Tenant Architecture
2. UAA Setup
3. UAA Auth for Kubo
4. Kubo Roles
5. Kubo Quotas

1. Kubo Tenant Architecture (1/2)


User
Identity Provider
kubectl
API Server
Login to IdP
IdP provide access_token and id_token
Call kubectl using provided id_token
Send token in Authorization header to the API server
Validate JWT signature
Check id_token expiration date
User Authorized?
Send response to kubectl
Send result to the user

1. Kubo Tenant Architecture (2/2)

Model 1


1
UAA User Management Server
Total Resources
2
Central K8's Clusters Administrator
Service Account
3
Quota
K8's Cluster Resource Quota Allocation
2
User 1 ~ N
Role
Namespace Tenant

2. UAA Setup (1/5)

- UAA-Release (1/5)

Note: This deployment Guide is targeted only Ubuntu 16.04 users.

Note: Perform below steps in your Inception.

- ssh into inception

```
$ ssh -i your-key ubuntu@yours-instance-public-ip
$ cd ~/workspace
```

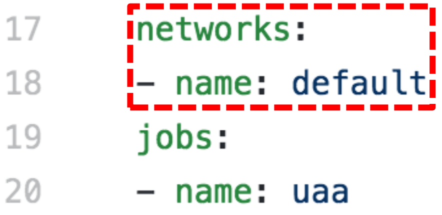
- Next clone the bosh-deployment and kubo-deployment repositories

```
$ git clone https://github.com/cloudfoundry/uaa-release.git -b v0.60.2
$ cd ~/ workspace/uaa-release/templates/
```

- Edit uaa-deployment.yml file to configure public ip.

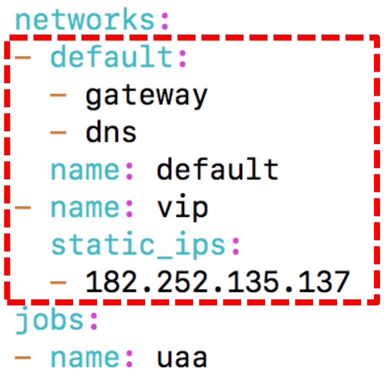
```
$ vi ~/workspace/uaa-release/templates/uaa-deployment.yml
```

Before Editing



```
networks:
- name: default
jobs:
- name: uaa
```

After Editing



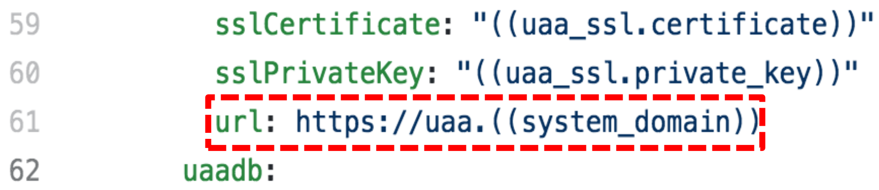
```
networks:
- default:
  - gateway
  - dns
  name: default
- name: vip
  static_ips:
  - 182.252.135.137
jobs:
- name: uaa
```

2. UAA Setup (2/5)
- UAA-Release (2/5)

❖ Edit uaa-deployment.yml file to configure uaa public ip to uaa-url to access public.

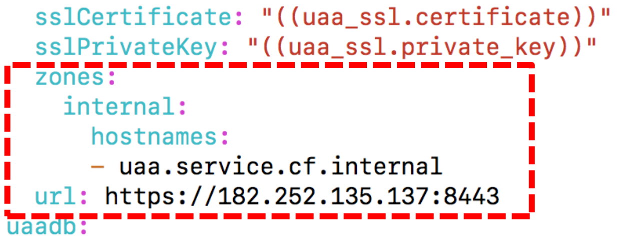
```
$ vi ~/workspace/uaa-release/templates/uaa-deployment.yml
```

Before Editing



```
  sslCertificate: "((uaa_ssl.certificate))"
  sslPrivateKey: "((uaa_ssl.private_key))"
  url: https://uaa.((system_domain))
uaadb:
```

After Editing

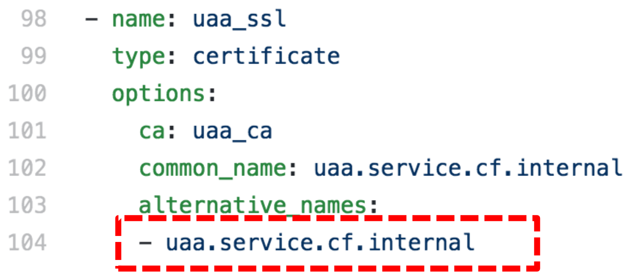


```
  sslCertificate: "((uaa_ssl.certificate))"
  sslPrivateKey: "((uaa_ssl.private_key))"
  zones:
    internal:
      hostnames:
      - uaa.service.cf.internal
  url: https://182.252.135.137:8443
uaadb:
```

❖ Edit uaa-deployment.yml file to configure public ip and Kubo LB IP server in certificates.

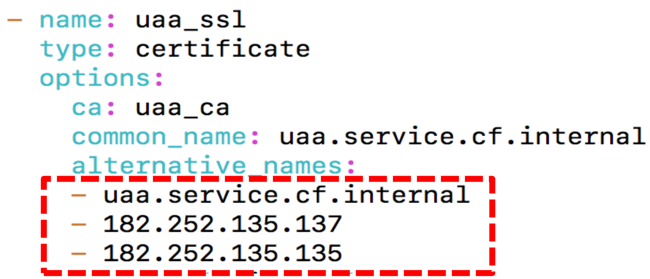
```
$ vi ~/workspace/uaa-release/templates/uaa-deployment.yml
```

Before Editing



```
- name: uaa_ssl
  type: certificate
  options:
    ca: uaa_ca
    common_name: uaa.service.cf.internal
    alternative_names:
    - uaa.service.cf.internal
```

After Editing



```
- name: uaa_ssl
  type: certificate
  options:
    ca: uaa_ca
    common_name: uaa.service.cf.internal
    alternative_names:
    - uaa.service.cf.internal
    - 182.252.135.137
    - 182.252.135.135
```

2. UAA Setup (3/5)

- UAA-Deployment & Uaa Users (3/5)

Note: Use Kubo Cloud Config for UAA deployment. Make sure to match vm_type, networks & persistent_disk_type.

❖ Deploy Uaa standalone.

```
$ bosh -e kubo –d uaa deploy ~/workspace/uaa-release/templates/uaa-deployment.yml
```

Note: Download UAA Certs, Password from credhub as illustrated in Kubo deployment guide.

Note: If you have CF UAA you can target and create user and client .

❖ Target Standalone UAA Server Using uaac cli.

```
$ uaac target https://your-uaa-public-ip:8443 --skip-ssl-validation
$ uaac token client get admin -s your-uaa_admin_client_secret
```

❖ Target CF UAA Server Using uaac cli.

```
$ uaac target https://uaa.your-domain --skip-ssl-validation
$ uaac token client get admin -s your-uaa_admin_client_secret
```

❖ Add User in targeted UAA Server.

```
$ uaac user add your-user-name --given_name kubo --family_name kubernetes --email Your-Email
--origin uaa -p your-user-password
```

❖ List all users from UAA Server Using uaac cli.

```
$ uaac users
```

❖ Check your user is registered in UAA Server Using uaac cli.

```
$ uaac user get your-user-name
```

2. UAA Setup (4/5)

- UAA-Clients (4/5)

❖ Create uaa client to configure kubo.

```
$ cd ~/workspace/uaa-release/templates/ && mkdir kubo && cd ~/workspace/uaa-release/templates/kubo
$ vi kubo-uaa.sh
```

```
#!/bin/bash

# Clinet Info
ADMIN=admin
ADMIN_SECRET=your-uaa_admin_client_secret
CLIENT=your-client-name
CLIENT_SECRET=your-client-secret
AUTHORITIES="oauth.login,scim.write,clients.read,scim.userids,password.write,clients.secret,clients.write,uaa.admin,scim.read,doppler.firehose"
AUTHORIZED_GRANT_TYPES="client_credentials,password,refresh_token" # authorization_code,implicit
SCOPE="cloud_controller.read,cloud_controller.write,openid,cloud_controller.admin,scim.read,scim.write,doppler.firehose,uaa.user,routing.router_groups.read,uaa.admin,password.write"

# [2] Add Client
uaac target https://your-uaa-public-ip:8443 --skip-ssl-validation
uaac token client get $ADMIN -s $ADMIN_SECRET
uaac client add $CLIENT --name $CLIENT -s $CLIENT_SECRET ₩
--authorities $AUTHORITIES ₩
--authorized_grant_types $AUTHORIZED_GRANT_TYPES ₩
--scope $SCOPE
```

❖ Grant permissions for kubo-uaa.sh file and create client in uaa setver for kubo dpeloyment.

```
$ chmod 777 ~/workspace/uaa-release/templates/kubo/kubo-uaa.sh
$ ./kubo-uaa.sh
```

❖ List all clients from UAA Server Using uaac cli.

```
$ uaac clients
```

❖ Check your user is registered in UAA Server Using uaac cli.

```
$ uaac client get your-client-name
```

2. UAA Setup (5/5)

- UAA-Config (5/5)

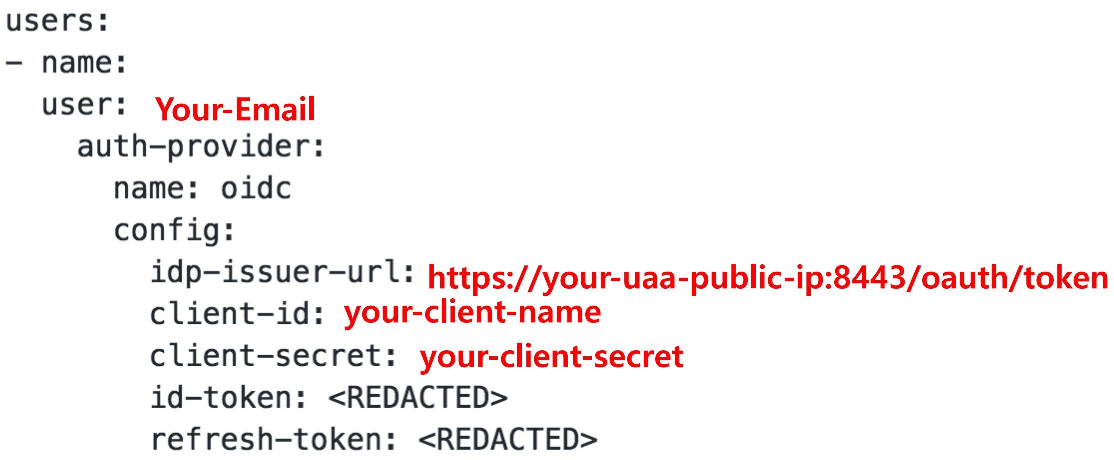
- Install uaak cli to create uaa config.

```
$ cd ~/workspace/
$ git clone https://github.com/abhilash07/uaa.git  && cd ~/workspace/uaa
$ chmod +x uaak
$ sudo mv ~/workspace/uaa/uaak /usr/local/bin/uaak
$ uaak -h
```

- Create uaa config to configure kubo config using uaak cli.

```
$ uaak -uaa.username=Your-Email -uaa.password=your-user-password -uaa.client_id=your-client-name -uaa.client_secret=your-client-secret -uaa.url=https://your-uaa-public-ip:8443 -uaa.skip_ssl_verify=true >> ~/workspace/uaa-release/templates/kubo/uaa-kubo.json
```

Sample Output



```
users:
- name:
  user:  Your-Email
    auth-provider:
      name: oidc
      config:
        idp-issuer-url: https://your-uaa-public-ip:8443/oauth/token
        client-id: your-client-name
        client-secret:  your-client-secret
        id-token: <REDACTED>
        refresh-token: <REDACTED>
```

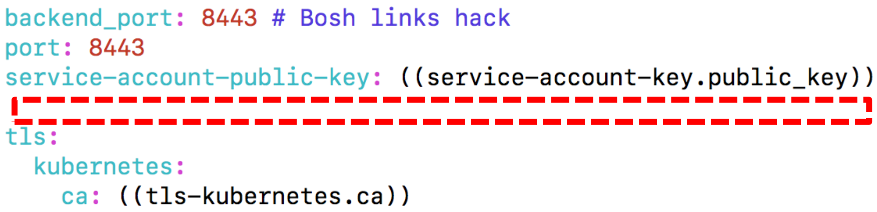
3. UAA Auth kubo (1/4)
- Kubo OAuth (1/4)

Note: Make Sure to have Kubo deployment environment. If not please follow deployment guide to deploy on y our preferred IaaS.

❖ Edit kubo-deployment cfcr.yml file for configuring uaa.

```
$ vi ~/workspace/kubo-deployment/manifests/cfcr.yml
```

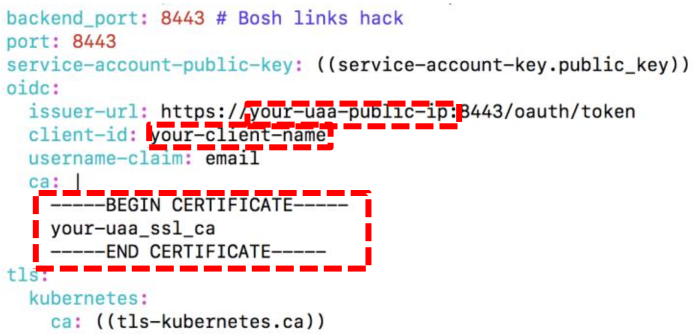
Before Editing



```
backend_port: 8443 # Bosh links hack
port: 8443
service-account-public-key: ((service-account-key.public_key))

tls:
  kubernetes:
    ca: ((tls-kubernetes.ca))
```

After Editing



```
backend_port: 8443 # Bosh links hack
port: 8443
service-account-public-key: ((service-account-key.public_key))
oidc:
  issuer-url: https://your-uaa-public-ip:8443/oauth/token
  client-id: your-client-name
  username-claim: email
  ca: |
    -----BEGIN CERTIFICATE-----
    your-uaa_ssl_ca
    -----END CERTIFICATE-----
tls:
  kubernetes:
    ca: ((tls-kubernetes.ca))
```

❖ Update your cfcr deployment on your preferred IaaS, to configures uaa with kubernetes api-server.

3. UAA Auth kubo (2/4)
- Kubo config (2/4)

❖ Update Kubo config in ~/.kube/config using kubectl cli. Get tokens and other values from uaa-kubo.json.

Note: Get uaa-kubo.json file form section 2 UAA Setup (5/5) - UAA-Config (5/5) .

```
kubectl config set-credentials Your-Email --auth-provider=oidc --auth-provider-arg=idp-issuer-url=https://your-uaa-public-ip:8443/oauth/token --auth-provider-arg=client-id=your-client-name --auth-provider-arg=client-secret=your-client-secret --auth-provider-arg=refresh-token=yourclinet-id-token --auth-provider-arg=idp-certificate-authority=your-uaa_ssl.crt --auth-provider-arg=id-token=your-refresh-token
```

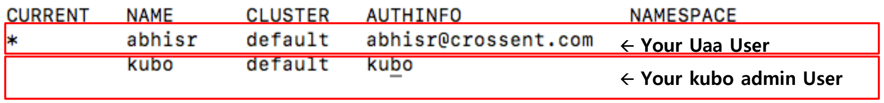
❖ Configure k8s cluster context with you above uaa user.

```
$ kubectl config set-context <context-name> --cluster <cluster-name> --user <Your-Email>
$ kubectl config use-context <context-name>
```

❖ List your k8s cluster contexts.

```
$ kubectl config get-contexts
```

Sample Output



```
CURRENT   NAME     CLUSTER   AUTHINFO              NAMESPACE
*         abhisr   default   abhisr@crossent.com
          kubo     default   kubo
```

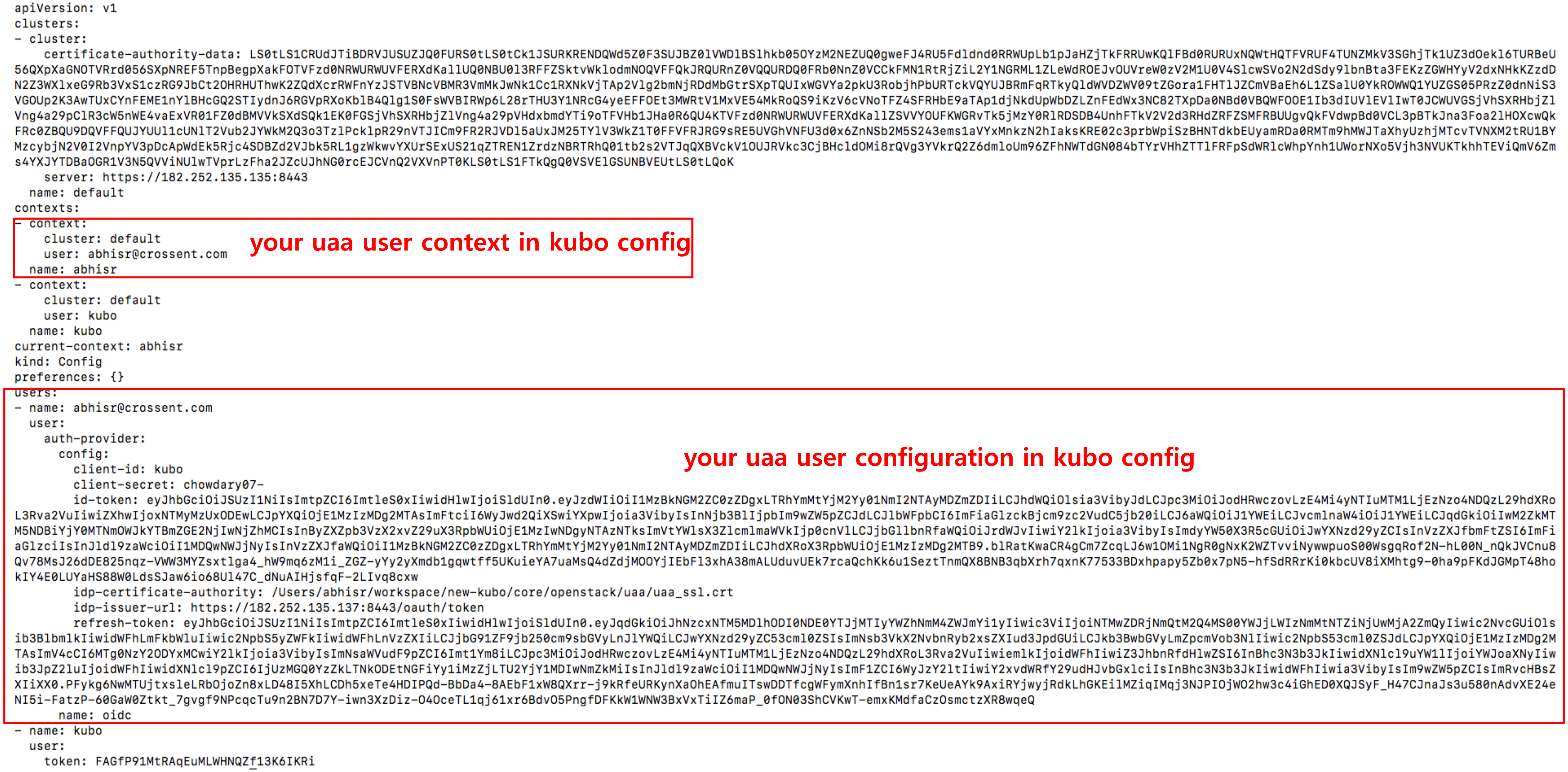
← Your Uaa User

← Your kubo admin User

Note: * in above screenshot represents the current context of your k8s cluster

3. UAA Auth kubo (3/4)
- Kubo config (3/4)

Sample Output of kubo config after configuring your uaa user and client



```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURKRENDQWd5Z0F3SUJBZ0lVWDlBSlhkb05OYzM2NEZUQ0gweFJ4RU5Fdldnd0RRWUpLb1pJaHZjTkFRRUwKQlFBd0RURUxNQWtHQTFVRUF4TUNZMkV3SGhjTk1UZ3dOekl6TURBeU
56QXpXaGNOTVRrd056SXpNREF5TnpBegpXakFOTVFzd0NRWURWUVFERXdKallUQ0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCCkFMN1RtRjZiL2Y1NGRML1ZLeWdROEJvOUVreW0zV2M1U0V4SlcwSVo2N2dSdy9lbnBta3FEKzZGWHYyV2dxNHkKZzdD
N2Z3WXlxeG9Rb3VxS1czRG9JbCt2OHRHUThwK2ZQdXcrRWFnYzJSTVBNcVBMR3VmMkJwNk1Cc1RXNkVjTAp2Vlg2bmNjRDdMbGtrSXpTQUIxWGVYa2pkU3RobjhPbURTckVQYUJBRmFqRTkyQldWVDZWV09tZGora1FHTlJZCmVBaEh6L1ZSalU0YkROWWQ1YUZGS05PRzZ0dnNiS3
VGOUp2K3AwTUxCYnFEME1nYlBHcGQ2STIydnJ6RGVpRXoKblB4Qlg1S0FsWVBIRWp6L28rTHU3Y1NRcG4yeEFFOEt3MWRtV1MxVE54MkRoQS9iKzV6cVNoTFZ4SFRHbE9aTAp1djNkdUpWbDZLZnFEdWx3NC82TXpDa0NBd0VBQWFOOE1Ib3dIUVlEVlIwT0JCWUVGSjVhSXRHbjZl
Vng4a29pClR3cW5nWE4vaExVR01FZ0dBMVVkSXdSQk1EK0FGSjVhSXRHbjZlVng4a29pVHdxbmdYTi9oTFVHb1JHa0R6QU4KTVFzd0NRWURWUVFERXdKallZSVVYOUFKWGRvTk5jMzY0RlRDSDB4UnhFTkV2V2d3RHdZRFZSMFRBUUgvQkFVdwpBd0VCL3pBTkJna3Foa2lHOXcwQk
FRc0ZBQU9DQVFFQUJYUUl1cUNlT2Vub2JYWkM2Q3o3TzlPcklpR29nVTJICm9FR2RJVDl5aUxJM25TY1V3WkZ1T0FFVFRJRG9sRE5UVGhVNFU3d0x6ZnNSb2M5S243ems1aVYxMnkzN2hIaksKRE02c3prbWpiSzBHNTdkbEUyamRDa0RMTm9hMWJTaXhyUzhjMTcvTVNXM2tRU1BY
MzcybjN2V0I2VnpYV3pDcApWdEk5Rjc4SDBZd2VJbk5RL1gzWkwvYXUrSExUS21qZTREN1ZrdzNBRTRhQ01tb2s2VTJqQXBVckV1OUJRVkc3CjBHcldOMi8rQVg3YVkrQ2Z6dmloUm96ZFhNWTdGN084bTYrVHhZTTlFRFpSdWRlcWhpYnh1UWorNXo5Vjh3NVUKTkhhTEViQmV6Zm
s4YXJYTDBaOGR1V3N5QVViNUlwTVprLzFha2JZcUJhNG0rcEJCVnQ2VXVnPT0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQoK
    server: https://182.252.135.135:8443
  name: default
contexts:
- context:
    cluster: default
    user: abhisr@crossent.com
  name: abhisr
- context:
    cluster: default
    user: kubo
  name: kubo
current-context: abhisr
kind: Config
preferences: {}
users:
- name: abhisr@crossent.com
  user:
    auth-provider:
      config:
        client-id: kubo
        client-secret: chowdary07-
        id-token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImtleS0xIiwidHlwIjoiSldUIn0.eyJzdWIiOiI1MzBkNGM2ZC0zZDgxLTRhYmMtYjM2Yy01NmI2NTAyMDZmZDIiLCJhdWQiOlsia3VibyJdLCJpc3MiOiJodHRwczovLzE4Mi4yNTIuMTM1LjEzNzo4NDQzL29hdXRo
L3Rva2VuIiwiZXhwIjoxNTMyMzUxODEwLCJpYXQiOjE1MzIzMDg2MTAsImFtciI6WyJwd2QiXSwiYXpwIjoia3VibyIsInNjb3BlIjpbIm9wZW5pZCJdLCJlbWFpbCI6ImFiaGlzckBjcm9zc2VudC5jb20iLCJ6aWQiOiJ1YWEiLCJvcmlnaW4iOiJ1YWEiLCJqdGkiOiIwM2ZkMT
M5NDBiYjY0MTNmOWJkYTBmZGE2NjIwNjZhMCIsInByZXZpb3VzX2xvZ29uX3RpbWUiOjE1MzIwNDgyNTAzNTksImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJjbGllbnRfaWQiOiJrdWJvIiwiY2lkIjoia3VibyIsImdyYW50X3R5cGUiOiJwYXNzd29yZCIsInVzZXJfbmFtZSI6ImFi
aGlzciIsInJldl9zaWciOiI1MDQwNWJjNyIsInVzZXJfaWQiOiI1MzBkNGM2ZC0zZDgxLTRhYmMtYjM2Yy01NmI2NTAyMDZmZDIiLCJhdXRoX3RpbWUiOjE1MzIzMDg2MTB9.blRatKwaCR4gCm7ZcqLJ6w1OMi1NgR0gNxK2WZTvviNywwpuoS00WsgqRof2N-hL00N_nQkJVCnu8
Qv78MsJ26dDE825nqz-VWW3MYZsxtlga4_hW9mq6zM1i_ZGZ-yYy2yXmdb1gqwtff5UKuieYA7uaMsQ4dZdjMOOYjIEbFl3xhA38mALUduvUEk7rcaQchKk6u1SeztTnmQX8BNB3qbXrh7qxnK77533BDxhpapy5Zb0x7pN5-hfSdRRrKi0kbcUV8iXMhtg9-0ha9pFKdJGMpT48ho
kIY4E0LUYaHS88W0LdsSJaw6io68Ul47C_dNuAIHjsfqF-2LIvq8cxw
        idp-certificate-authority: /Users/abhisr/workspace/new-kubo/core/openstack/uaa/uaa_ssl.crt
        idp-issuer-url: https://182.252.135.137:8443/oauth/token
        refresh-token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImtleS0xIiwidHlwIjoiSldUIn0.eyJqdGkiOiJhNzcxNTM5MDlhODI0NDE0YTJjMTIyYWZhNmM4ZWJmYi1yIiwic3ViIjoiNTMwZDRjNmQtM2Q4MS00YWJjLWIzNmMtNTZiNjUwMjA2ZmQyIiwic2NvcGUiOls
ib3BlbmlkIiwidWFhLmFkbWluIiwic2NpbS5yZWFkIiwidWFhLnVzZXIiLCJjbG91ZF9jb250cm9sbGVyLnJlYWQiLCJwYXNzd29yZC53cml0ZSIsImNsb3VkX2NvbnRyb2xsZXIud3JpdGUiLCJkb3BwbGVyLmZpcmVob3NlIiwic2NpbS53cml0ZSJdLCJpYXQiOjE1MzIzMDg2M
TAsImV4cCI6MTg0NzY2ODYxMCwiY2lkIjoia3VibyIsImNsaWVudF9pZCI6Imt1Ym8iLCJpc3MiOiJodHRwczovLzE4Mi4yNTIuMTM1LjEzNzo4NDQzL29hdXRoL3Rva2VuIiwiemlkIjoidWFhIiwiZ3JhbnRfdHlwZSI6InBhc3N3b3JkIiwidXNlcl9uYW1lIjoiYWJoaXNyIiw
ib3JpZ2luIjoidWFhIiwidXNlcl9pZCI6IjUzMGQ0YzZkLTNkODEtNGFiYy1iMzZjLTU2YjY1MDIwNmZkMiIsInJldl9zaWciOiI1MDQwNWJjNyIsImF1ZCI6WyJzY2ltIiwiY2xvdWRfY29udHJvbGxlciIsInBhc3N3b3JkIiwidWFhIiwia3VibyIsIm9wZW5pZCIsImRvcHBsZ
XIiXX0.PFykg6NwMTUjtxsleLRbOjoZn8xLD48I5XhLCDh5xeTe4HDIPQd-BbDa4-8AEbF1xW8QXrr-j9kRfeURKynXaOhEAfmuITswDDTfcgWFymXnhIfBn1sr7KeUeAYk9AxiRYjwyjRdkLhGKEilMZiqIMqj3NJPIOjWO2hw3c4iGhED0XQJSyF_H47CJnaJs3u580nAdvXE24e
NI5i-FatzP-60GaW0Ztkt_7gvgf9NPcqcTu9n2BN7D7Y-iwn3XzDiz-O4OceTL1qj61xr6BdvO5PngfDFKkW1WNW3BxVxTiIZ6maP_0fON03ShCVKwT-emxKMdfaCzOsmctzXR8wqeQ
      name: oidc
- name: kubo
  user:
    token: FAGfP91MtRAqEuMLWHNQZf13K6IKRi
```

your uaa user context in kubo config

your uaa user configuration in kubo config

3. UAA Auth kubo (4/4)
- Kubo config (4/4)

Note: To utilize resources from k8s cluster request your administrator to create ClusterRole or Role to register your Uaa User to K8s api-server .

❖ Change the k8s admin context to create ClusterRole and Role.

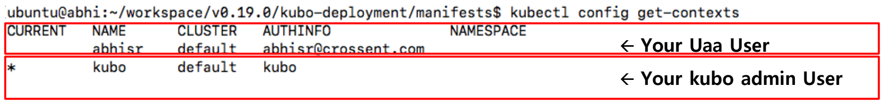
```
$ kubectl config use-context kubo
```

Note: After changing to k8s admin context, next section explain how to create ClusterRole and Role to register your Uaa User to K8s api-server.

❖ List your k8s cluster contexts.

```
$ kubectl config get-contexts
```

Sample Output



```
ubuntu@abhi:~/workspace/v0.19.0/kubo-deployment/manifests$ kubectl config get-contexts
CURRENT   NAME      CLUSTER   AUTHINFO              NAMESPACE
          abhisr    default   abhisr@crossent.com
*         kubo      default   kubo
```

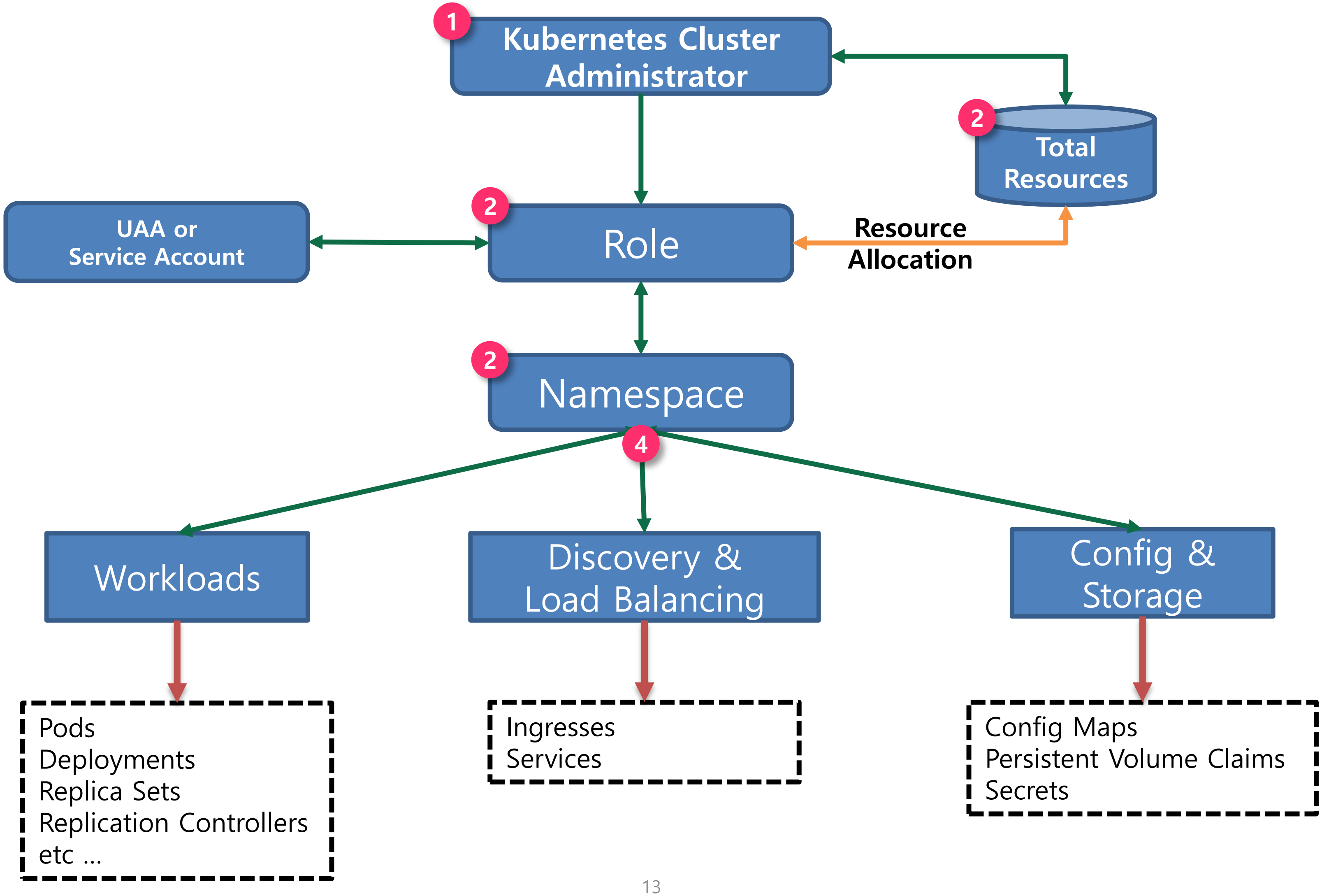
← Your Uaa User

← Your kubo admin User

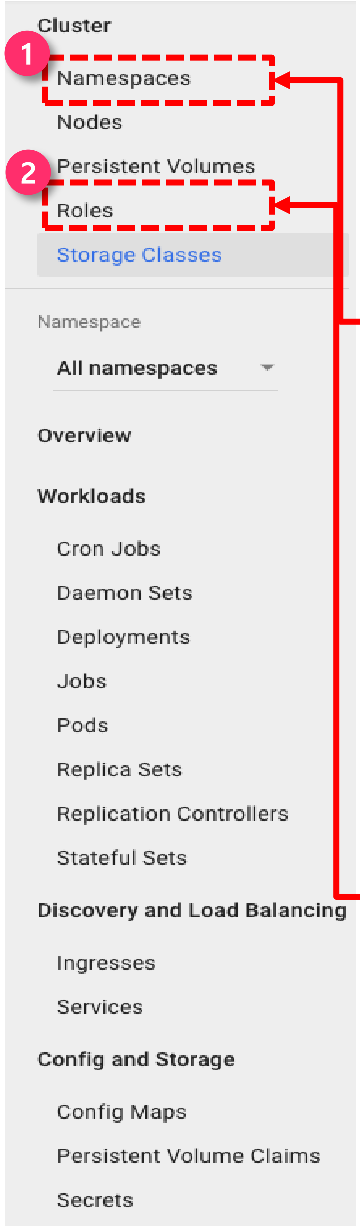
Note: * in above screenshot represents the current context of your k8s cluster

4. Kubo Roles (1/2)
- Role Architecture

Role, controls the entire resources allocated to uaa user and also controls all the objects with in the **namespace** only.



4. Kubo Roles (2/2)
- Role Deployment



Note: In this tutorial we enable all permissions for Role user. For limited permissions follow this link. Role

1. Create Namespace

```
$ kubectl create –f namespace.json
```

namespace.json

```
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "abhi"
  }
}
```

Role: In the RBAC API, a role contains rules that represent a set of permissions. Permissions are purely additive (there are no “deny” rules). A role can be defined within a **namespace** with a **Role**.

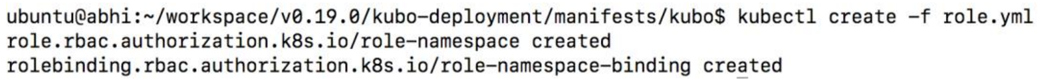
2. Create Role

```
$ kubectl create –f role.yml
```

role.yml

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: abhi
  name: role-namespace
rules:
  - apiGroups: ["*"]
    resources: ["*"]
    verbs: ["*"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: role-namespace-binding
  namespace: abhi
subjects:
  - kind: User
    name: abhisr@crossent.com
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: role-namespace
  apiGroup: rbac.authorization.k8s.io
```

Sample Output



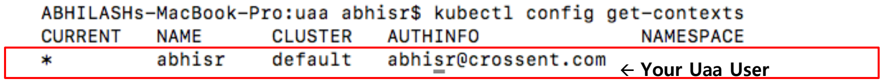
```
ubuntu@abhi:~/workspace/v0.19.0/kubo-deployment/manifests/kubo$ kubectl create -f role.yml
role.rbac.authorization.k8s.io/role-namespace created
rolebinding.rbac.authorization.k8s.io/role-namespace-binding created
```

2. Now we can start deploying service and pods at **Namespace** by using above **Role**.

3. Change to tenant context to start using **namespace** resources and services.

```
$ kubectl config use-context abhisr
```

Sample Output



```
ABHILASHs-MacBook-Pro:uaa abhisr$ kubectl config get-contexts
CURRENT   NAME      CLUSTER   AUTHINFO              NAMESPACE
*         abhisr    default   abhisr@crossent.com
```

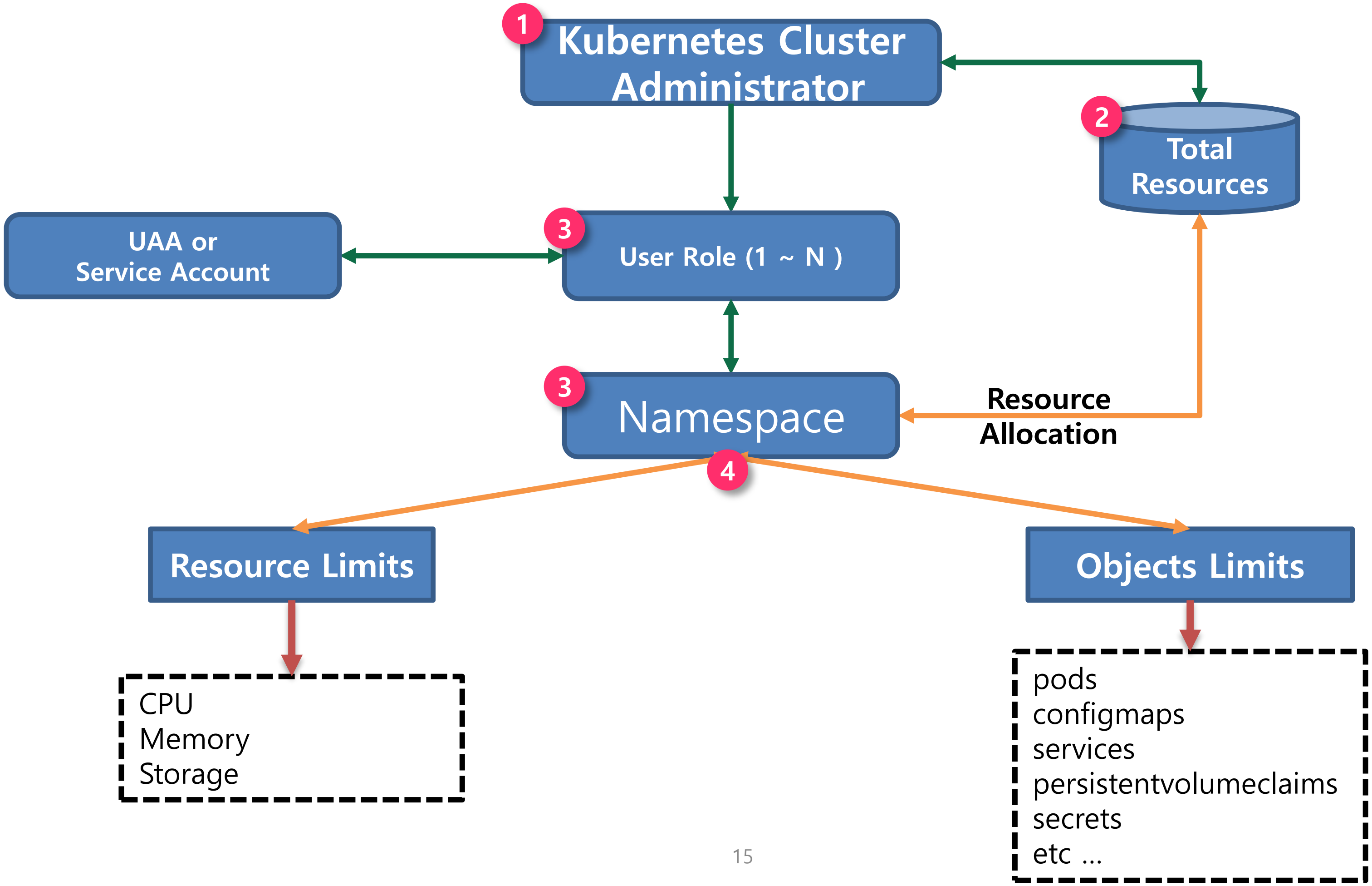
← Your Uaa User

5. Kubo Quotas (1/4)
- Resource Quota

- When Kubernetes cluster is used by multiple people or teams, resource management becomes more important.
- You Want to be able to manage the resources you give to a person or a team. Because you don't want one person or team taking up all the resources.
- In this tutorial Resource Quota divided into two parts, 1) Resource Limits & 2) Object Limits.

1) Resource Limits: Administrator can set CPU, Memory and storage limitation for namespace tenant.

2) Object Limits: Administrator can set Pods, configmaps, services, persistentvolumeclaims, replicationcontrollers, secrets etc. limitation for namespace tenant.



5. Kubo Quotas (2/4)

- Resource Quota

Note: Please Create Namespace and Role before creating Resource Quota by following 4. Kubo Roles Section.

1. Create Resource Limits

```
$ kubectl create –f resource-limit.yml
```

resource-limit.yml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-quota
  namespace: tenant
spec:
  hard:
    pods: “4”
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

Resource Limits: Administrator can set CPU, Memory and storage limitation for namespace tenant.

2. Create Object Limits

```
$ kubectl create –f object-limit.yml
```

object-limit.yml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-quota
  namespace: tenant
spec:
  hard:
    configmaps: "10"
    persistentvolumeclaims: "4"
    replicationcontrollers: "20"
    secrets: "10"
    services: "10"
    services.loadbalancers: "2"
```

Object Limits: Administrator can set Pods, configmaps, services, persistentvolumeclaims, replicationcontrollers, secrets etc. limitation for namespace tenant.

5. Kubo Quotas (3/4)
- Resource Quota Deployment

Note: Please Create resource-limit and object-limit by following 5. Kubo Quotas → Resource Quota Section.

1. Create deployment with quotas

   $ kubectl create –f deployment-with-quotas.yml

deployment-with-quotas.yml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: helloworld-deployment
  namespace: tenant
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
      - name: k8s-demo
        image: wardviaene/k8s-demo
        ports:
        - name: nodejs-port
          containerPort: 3000
        resources:
          requests:
            cpu: 200m
            memory: 0.5Gi
          limits:
            cpu: 400m
            memory: 1Gi
```

❖ Each container can specify request capacity and capacity limits.

❖ **Request capacity** is an explicit requests for resources, we can this it as **minimum amount of resource the pod needs**.

❖ **Resource limit** is a limit imposed to the container, it means the **container will not be able to utilize more resources than specified**.

❖ You run a deployment with a pod with a CPU resource request of 200m.

❖ Here 200m = 200 millicpu (also known as 200 millicores). 200m = 0.2 which is 20% of a CPU core of a running worker nodes. For example if you have 2 Cores, it’s still 20% usage from our total cores.

❖ Memory quotas are defined by MiB or GiB.

5. Kubo Quotas (4/4)
- Resource Default Quota

Note: Please Create Namespace and Role before creating Resource Default Quota by following 4. Kubo Roles Section.

1. Create Resource Limits

```
$ kubectl create –f default.yml
```

default.yml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limits
  namespace: tenant
spec:
  limits:
  - default:
      cpu: 200m
      memory: 512Mi
    defaultRequest:
      cpu: 100m
      memory: 256Mi
    type: Container
```

Resource Default Quota: Some times user don't wants to mention quotas in container creation.

2. Create deployment with out quotas

```
$ kubectl create –f deployment-with-no-quotas.yml
```

deployment-with-no-quotas.yml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: helloworld-deployment
  namespace: tenant
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: helloworld
    spec:
      containers:
      - name: k8s-demo
        image: wardviaene/k8s-demo
        ports:
        - name: nodejs-port
          containerPort: 3000
```

THANK YOU