0404 - Debug Pods - Create Org Using Local Pod (or local git bash)

Objective

To use a debug pod (or local git bash) to:

- Check server status
- Create Org
- List Orgs

Method

Start Alpine Debug Pod

Create interactive shell session with a new temporary debug pod

> Use kubectl to create debug alpine pod (assumes you are in the right context etc)
>
> ```
> kubectl -n cluedin run -i --tty --rm debug --image=alpine --restart=Never
> ```
>
> You are now connected to the pod and can run commands

Install curl and jq

> in the pod interactive shell session
>
> ```
> / # apk --no-cache add curl jq
> ```

Get cluedin server internal name for the service on port 9000

> in powershell
>
> ```
> PS > kubectl -n cluedin get services | FindStr 9000
> cluedin-demo-cluedin                    ClusterIP      10.0.68.113    <none>
> 9000/TCP,9001/TCP,9003/TCP,9006/TCP,9007/TCP,9013/TCP   7d10h
> ```
>
> In this case `cluedin-demo-cluedin` is the service name that is running on port 9000, yours will vary depending on what names were given during creation of the cluster. This is the cluster name of the host.

In the pod interactive shell session - test access to cluedin server

> ```
> / # curl http://cluedin-demo-cluedin:9000/health/liveness && echo
> OK
> ```
>
> test cluedin server status - expect all ServiceStatus to be Green… otherwise you need to fix your cluster before continuing
>
> ```
> / # curl http://cluedin-demo-cluedin:9000/status | jq
> ```
>
> ```
> {
>   "ServiceStatus": "Green",
>   "UpdatedDate": "2021-12-15T22:49:06.9713766+00:00",
>   "DataShards": [
>     {
>       "Type": "Blob",
>       "ServiceStatus": "Green"
>     },
>     {
>       "Type": "Configuration",
>       "ServiceStatus": "Green"
>     },
>     {
>       "Type": "Data",
>       "ServiceStatus": "Green"
>     },
>     {
>       "Type": "Search",
>       "ServiceStatus": "Green"
>     },
>     {
>       "Type": "Graph",
> ```

```
            "ServiceStatus": "Green"
          },
          {
            "Type": "Metrics",
            "ServiceStatus": "Green"
          }
        ],
        "Components": [
          {
            "Type": "Api",
            "ServiceStatus": "Green"
          },
          {
            "Type": "Authentication",
            "ServiceStatus": "Green"
          },
          {
            "Type": "Crawling",
            "ServiceStatus": "Green"
          },
          {
            "Type": "Scheduling",
            "ServiceStatus": "Green"
          },
          {
            "Type": "ServiceBus",
            "ServiceStatus": "Green"
          },
          {
            "Type": "System",
            "ServiceStatus": "Green"
          }
        ]
      }
      / #
```


New Account Access Key

Note: Please be aware that your k8s cluster should always have NEWACCOUNTACCESSKEY set - otherwise anyone can create orgs!!!

Update the values.yaml

```
application:
  …
  cluedin:
    …
    roles:
      main:
        extraConfiguration:
          # remove most restrictions around org names (e.g. "cluedin")
          CLUEDIN_appSettings__ReservedOrganizationIds: "none"
          # disable Fuzzy Entity Matching
          CLUEDIN_APPSETTINGS__CLUEPROCESSING_FUZZYENTITYMATCHING_ENABLED
: "false"
          # enable header x-cluedin-newaccountaccesskey for all new org
requests
          CLUEDIN_APPSETTINGS__NEWACCOUNTACCESSKEY:
"726e72ef92650c903eec<redacted>ced80da310c2daf2c46"
```

Create Org using Script in Debug Pod

Define Environment

```
define these 2 environment variable exports as required

e.g:
$ export org=myorg
$ export password=bNaye2#06rU
```

Create script

Create this script with vi
or copy and paste into the console (skip the `#!/bin/sh` line if copy and paste)

Update the highlighted parts to match your app endpoint.
Note: replace **NewAccountAccessKey** with the correct key
Note: the cluster endpoint (and port number) may vary depending on your cluster config - if in doubt, use the public endpoint

create_org.sh

```
#!/bin/sh

URL=https://app.myk8scluster.com/auth/api/account/new
echo $URL

organizationName=$org
emailDomain=$org.com
username=admin@$emailDomain

curl -v -k --location -g --request POST $URL \
        --header 'Accept: application/json'\
        --header 'Content-Type: application/x-www-form-urlencoded'\
        --header 'x-cluedin-newaccountaccesskey: NewAccountAccessKey'\
        --data-urlencode email=$username \
        --data-urlencode username=$username \
        --data-urlencode password=$password \
        --data-urlencode confirmpassword=$password \
        --data-urlencode grant_type=password \
        --data-urlencode applicationSubDomain=$org \
        --data-urlencode organizationName=$organizationName \
        --data-urlencode emailDomain=$emailDomain \
        --data-urlencode allowEmailDomainSignup=true \
        | jq

URLHost=`echo $URL | sed -e 's|^[^/]*//||' -e 's|/.*$||'`
LoginHost=`echo $URLHost | sed -e "s|app|${org}|"`
URLSchema=`echo $URL | cut -d: -f 1`
SignInURL=$URLSchema://$LoginHost/signin

echo
echo $SignInURL
echo $username
echo $password
```

sample output:

```
{
  "Active": true,
  "ApplicationSubDomain": "myorg",
  "CustomerId": null,
  "ExternalAuthenticationId": null,
  "EmailDomainName": "myorg.com",
  "Id": "3226bffc-7c20-4cd2-921c-f32dd6ae1b38",
  "IsEmailDomainSignupActivated": true,
  "LastLoginTime": "2022-01-14T02:15:36.7961417Z",
  "Name": "cluedin",
  "Plan": "6749e7a5-da15-47e2-918b-6a385e288865",
  "PlanEndDate": "2022-02-13T02:15:36.7961418Z",
  "PlanIsActive": true,
  "PlanStartDate": "2022-01-14T02:15:36.7961422Z",
  "RefreshTokenLifeTime": 14400,
  "SubscriptionId": null
}

https://myorg.myk8scluster.com/signin
admin@myorg.com
bNaye2#06rU
```

Using git bash (instead of debug pod)

First time setup

run a git bash as admin
install jq

```
curl -L -o /usr/bin/jq.exe
https://github.com/stedolan/jq/releases/latest/download/jq-win64.exe
```

Using git bash

Then you can run the same curl script from above i.e. run same commands starting from "Define Environment" above

List all orgs using debug pod

Create debug pod for mssql commands

```
PS > kubectl run -i --tty --rm mssql-debug --image=mcr.microsoft.com/mssql-tools --
restart=Never -- sh
```

Download list_orgs

```
# curl -O
https://raw.githubusercontent.com/RudiBob/ImpTools/master/docker/ubuntu/list_orgs.sh
```

Update the script DBUSER and DBPASS with secrets / cluedin server pod environment:

```
# access local docker mssql
DBUSER='sa'
DBPASS='yourStrong(!)Password'
```

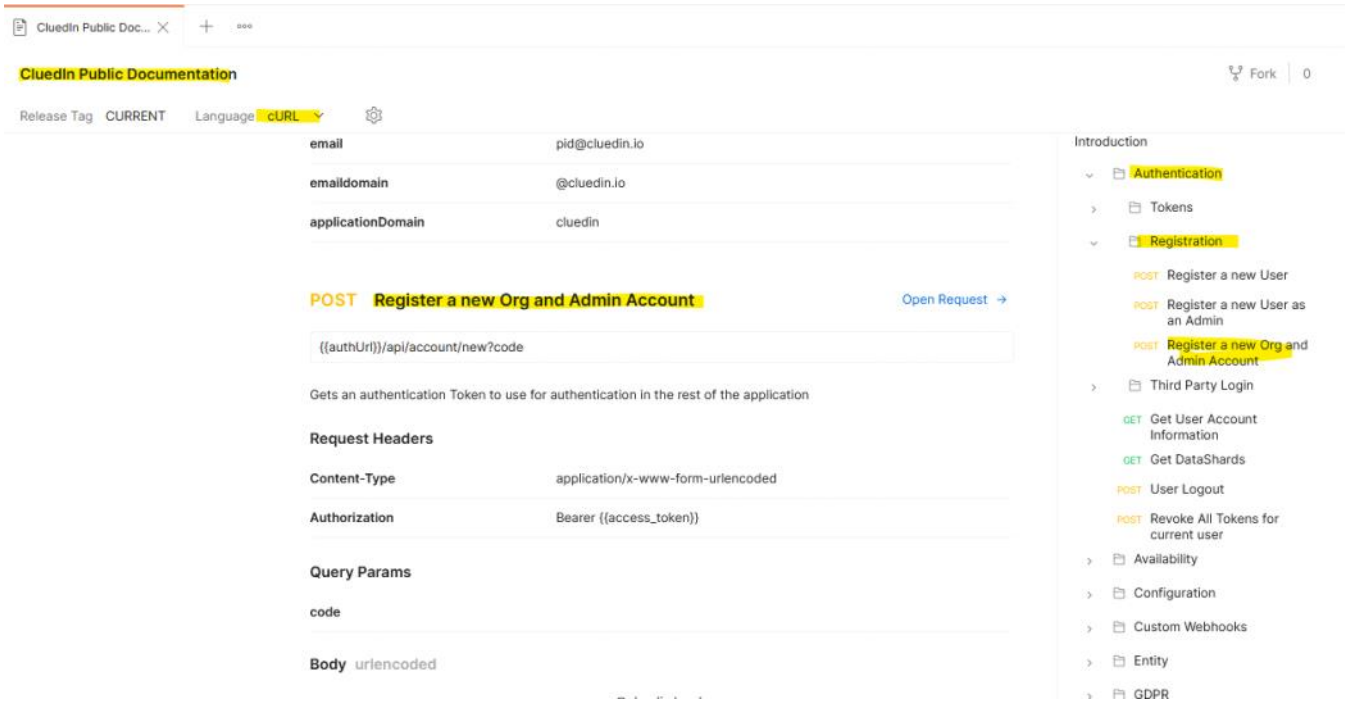
Run it - success!

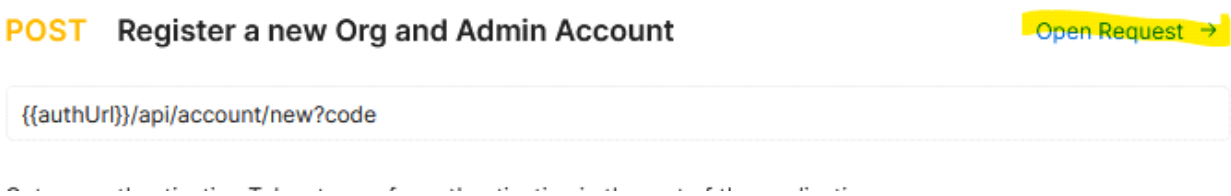
```
# ./list_orgs.sh
D376B0A8-6D88-487C-B6AA-2C6C96B765D9 cluedin    cluedin.com
860409C5-E63C-4F16-8652-54FDB35AB0C9 chetest7   chetest7.com
3C81D79B-F638-44A6-9C8C-736C0F31A8F2 chetest3   chetest3.com
86DD8F8F-7FEA-4EB8-B467-8B2CBDA2D2D7 chetest    chetest.com
50695A55-9CEA-4727-9B22-BBE90A68F62A chetest4   chetest4.com
7BD804BA-EDC3-4FEF-9B95-DFF6A0716CC6 chetest6   chetest6.com
3226BFFC-7C20-4CD2-921C-F32DD6AE1B38 chetest2   chetest2.com
0D227A99-D452-44C3-8578-F3754E873B9E chetest5   chetest5.com
```

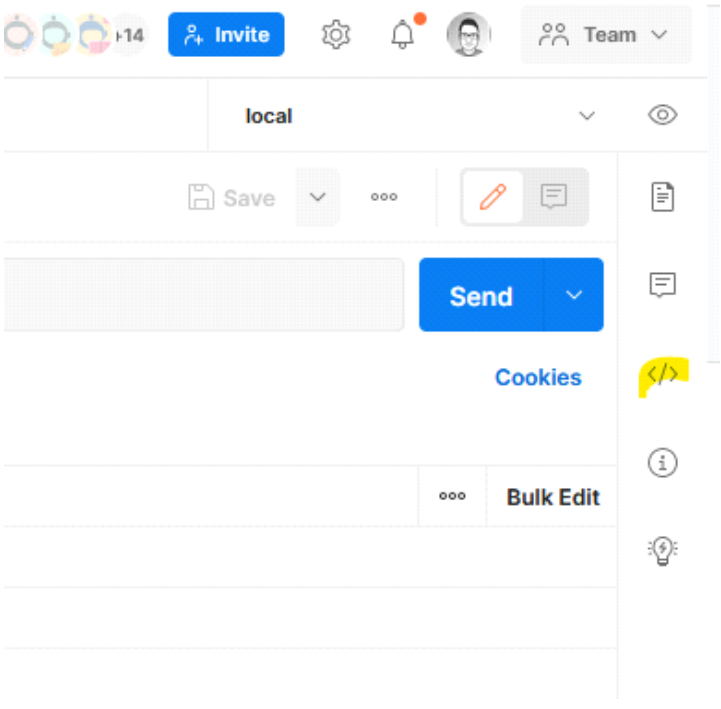
Postman References

Note: you can take postman API calls and run them as curl in the debug pod

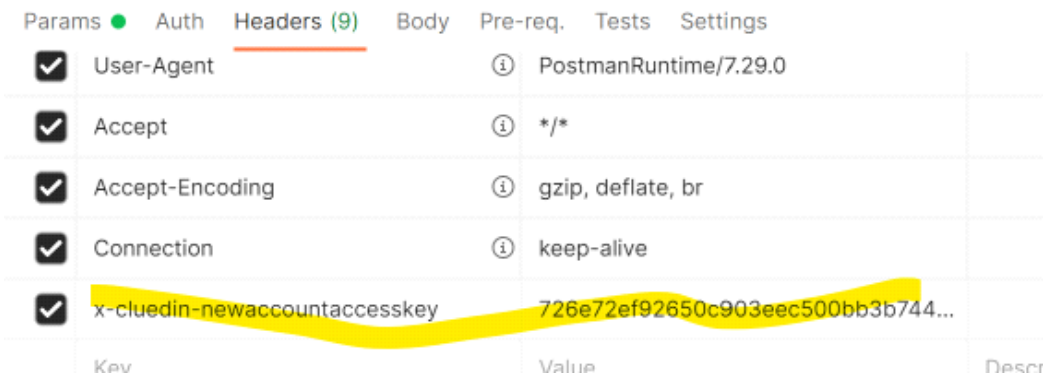
Create Org

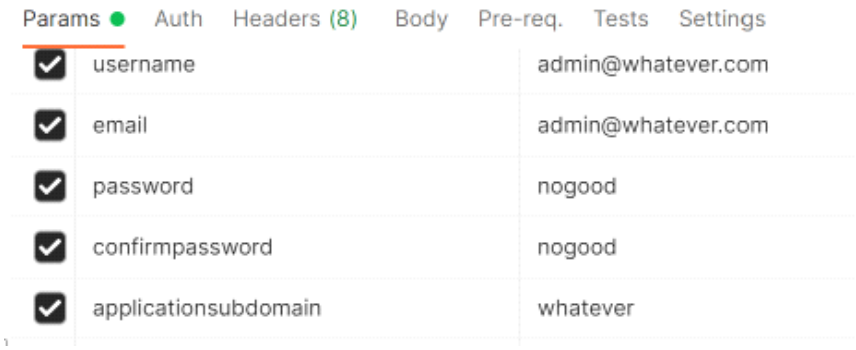


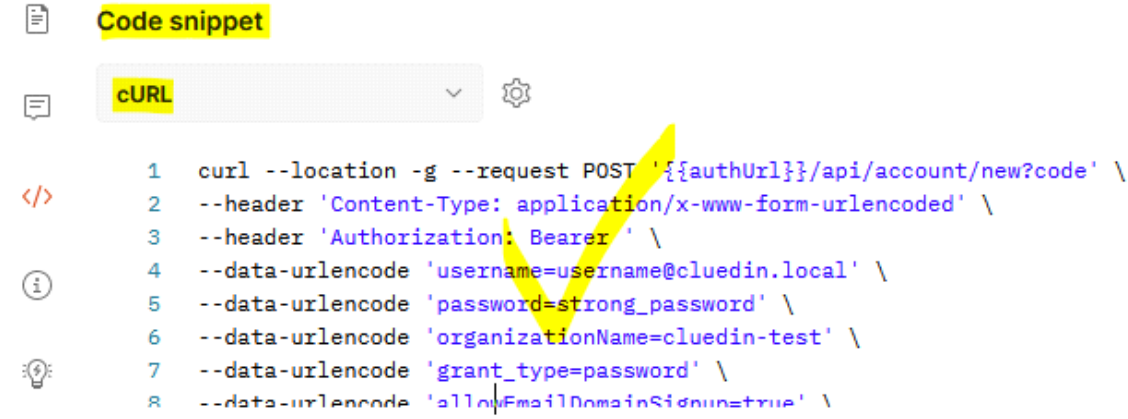




Note: the new account access key header


Params
Auth
Headers (9)
Body
Pre-req.
Tests
Settings
User-Agent
PostmanRuntime/7.29.0
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
x-cluedin-newaccountaccesskey
726e72ef92650c903eec500bb3b744...
Key
Value


Params
Auth
Headers (8)
Body
Pre-req.
Tests
Settings
username
admin@whatever.com
email
admin@whatever.com
password
nogood
confirmpassword
nogood
applicationsubdomain
whatever


Code snippet
cURL
curl --location -g --request POST '{{authUrl}}/api/account/new?code' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Bearer ' \
--data-urlencode 'username=username@cluedin.local' \
--data-urlencode 'password=strong_password' \
--data-urlencode 'organizationName=cluedin-test' \
--data-urlencode 'grant_type=password' \