# 1. Add users

See https://docs.openshift.com/container-platform/4.8/authentication/identity_providers/configuring-htpasswd-identity-provider.html.

OpenShift supports a variety of identity providers such as HTPasswd, Keystone, LDAP, Google etc. You will create an identity provider of type `htpasswd`, which allows defining users in a simple text file.

User lists are stored in a secret. The steps are: Obtain the current user list in htpasswd file form from the secret, add users, replace the secrect with the updated list.

## Display the identity provider

The CRC cluster is initialized with an htpasswd identity provider. Since identity providers are managed by OAuth, you must view the cluster's OAuth object.

You must have the cluster admin role.

```
$ oc login -u kubeadmin -p KUBEADMIN_PASSWD
$ oc get oauth
NAME      AGE
cluster   20d
$ oc get oauth cluster -o yaml
```

The interesting part is the spec, which should look like this:

```
spec:
  identityProviders:
  - htpasswd:
      fileData:
        name: htpass-secret
    mappingMethod: claim
    name: developer
    type: HTPasswd
  templates:
    login:
      name: login-template
  tokenConfig:
    accessTokenMaxAgeSeconds: 0
```

This OAuth object defines a single ID provider named *developer* (not related to the user account *developer*), which uses *htpasswd* to store users. The htpasswd file can be found in a secret named *htpass-secret*.

## Retrieve the current password file from the htpass-secret, manual method

The secret resides in a namespace named *openshift-config*.

```
$ oc get secret htpass-secret -n openshift-config -o json
{
    "apiVersion": "v1",
    "data": {
        "htpasswd":
"ZGV2ZWxvcGVyOiQyeSQwNSQxMW9TMFIxYUR4SFczdUJPSFBQeE1PRTJDdWZkMnpMSEVhZlNEakF3N2RNUHZMZUV5ajJFZQpiYmF1c2N
oOiQyeSQwNSRaQkpMUkNFS0xmLi5JeGJ1aWR6QUwuZkhnNThnMHFrNlQ0OEdUc2daZndvZUd3cm1zSmpKLgo="
    },
    "kind": "Secret",
    "metadata": {
...
```

Also try `-o yaml` instead of the JSON output format.

The secret's value is the base64-encoded content of the htpasswd file used to initialize the CRC cluster. It is the value of `data.htpasswd`, the orange text above. Copy that text, decode it and write it into a new file.

```
$ base64 -d > newusers.htpasswd <<<"ZGV2ZwxvcGVyOiQy..."
$ cat newusers.htpasswd
```

You should see two users, kubeadmin and developer, and their encrypted passwords.

## Retrieve the current password file from the htpass-secret, scriptable method

The manual method requires you to copy and paste text. The scriptable method is entirely automatic.

```
$ oc get secret htpass-secret -o jsonpath={.data.htpasswd} -n openshift-config |
  base64 -d > newusers.htpasswd
$ cat newusers.htpasswd
```

Here, you retrieve a subset of the JSON structure by providing the `-o jsonpath` option to the `get secret` command

## Add users

Add users *alice* and *bob*, with *openshift* as passwords. First, add a newline at the end of the htpasswd file.

```
$ echo >> newusers.htpasswd
$ htpasswd -bB newusers.htpasswd alice openshift
$ htpasswd -bB newusers.htpasswd bob openshift
$ htpasswd -bB newusers.htpasswd dummyuser openshift
$ cat newusers.htpasswd
```

## Replace the secret

View the YAML code of the new secret. The `dry-run=client` option has the effect that the `create secret` command does not create a new secret but only outputs the code. The secret is created from the htpasswd file you just crafted.

```
$ oc create secret generic htpass-secret --from-file=htpasswd=newusers.htpasswd --dry-run=client
-o yaml -n openshift-config > secret.yaml
$ cat secret.yaml
apiVersion: v1
data:
  htpasswd: ZGV2ZWxvcGVyOiQyYSQxMCRJa...
kind: Secret
metadata:
  creationTimestamp: null
  name: htpass-secret
  namespace: openshift-config
```

To replace the secret, use the manifest. However, rather than applying it, replace it, since you don't want an incremental change. You want no trace of the old secret data to remain.

```
$ oc replace -f secret.yaml
```

Double-check.

```
$ oc get secret htpass-secret -o jsonpath={.data.htpasswd} -n openshift-config | base64 -d
```

## Log on as the new users

List OpenShift users.

```
$ oc get user -n openshift-config
```

The new users don't exist in OpenShift yet. Right now, they are only in the htpasswd identity provider. To become user objects, they need to log on.

```
$ oc login -u alice -p openshift
$ oc login -u bob -p openshift
$ oc login -u dummyuser -p openshift
```

Display the `$HOME/.kube/config` file. You will find the three users together with tokens. These tokens, so-called bearer tokens, allow the users to log on to OAuth without providing a password. You can see your current bearer token this way:

```
$ oc whoami -t
```

List the users again.

```
$ oc get user -n openshift-config
Error from server (Forbidden): users.user.openshift.io is forbidden: User "dummyuser" cannot list
resource "users" in API group "user.openshift.io" at the cluster scope
```

Obviously, *dummyuser*'s roles don't allow it to list users. Log on as *kubeadmin* again (you may not require the password).

```
$ oc login -u kubeadmin -p KUBEADMIN_PASSWD
$ oc get user -n openshift-config
```

The new users should exist now.

# 2. User tasks

## Delete a user

Delete a user from the htpasswd file and update the secret again.

```
$ htpasswd -D newusers.htpasswd dummyuser
$ cat newusers.htpasswd
$ oc create secret generic htpass-secret --from-file=htpasswd=newusers.htpasswd --dry-run=client
-o yaml -n openshift-config > secret.yaml
$ oc replace -f secret.yaml
$ oc get user
NAME        UID                                    FULL NAME   IDENTITIES
...
dummyuser   8caa704b-fdcd-4946-b2e3-8cfab30b0434               developer:dummyuser
```

However, this just changes the secret and thereby the identity provider's data. *dummyuser* is still in OpenShift, and as long as it has a valid token, it can access the cluster:

```
$ oc login -u dummyuser
```

This still works. To avoid this, remove the OpenShift user object:

```
$ oc login -u kubeadmin
$ oc delete user dummyuser
$ oc login -u dummyuser
```

Now dummyuser is gone.

## Associate a user with a project

Give *bob* the basic role in project devproject.

```
$ oc adm policy add-role-to-user basic-user bob -n devproject
```

Do the same for alice, but use this manifest named alice-devproject.yaml:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: alice-devproject
  namespace: devproject
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: basic-user
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: alice
```

```
$ oc apply -f alice-devproject.yaml
```

Access the web console as kubeadmin, select **user management → users** and check success.