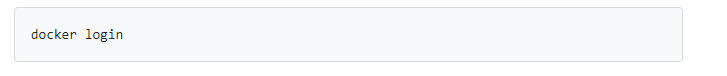
**Pull docker image from ECR**

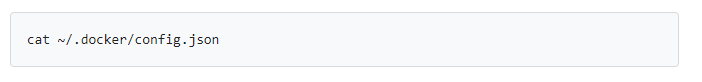
Login to docker



When prompted, enter your Docker username and password.

The login process creates or updates a config.json file that holds an authorization token.

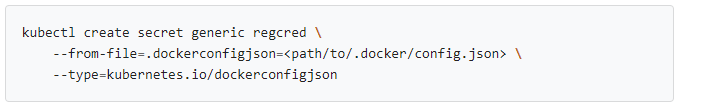
View the config.json file:



The output contains a section similar to this:

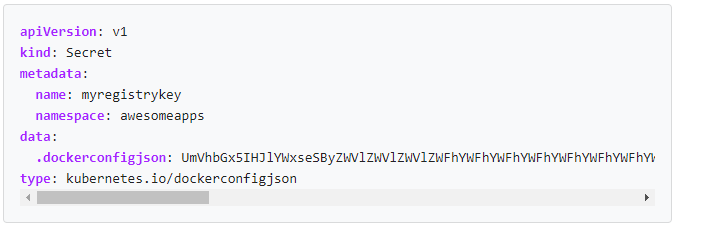


## Create a Secret based on existing Docker credentials



* set the name of the data item to .dockerconfigjson
* base64 encode the docker file and paste that string, unbroken as the value for field data[".dockerconfigjson"]
* set type to kubernetes.io/dockerconfigjson

create a secret yml file vi ecr-secret.yml



If you get the error message error: no objects passed to create, it may mean the base64 encoded string is invalid. If you get an error message like Secret "myregistrykey" is invalid: data[.dockerconfigjson]: invalid value ..., it means the base64 encoded string in the data was successfully decoded, but could not be parsed as a .docker/config.json file

## Create a Secret by providing credentials on the command line

kubectl create secret docker-registry regcred --docker-server=<your-registry-server> --docker-username=<your-name> --docker-password=<your-pword> --docker-email=<your-email>

* <your-registry-server> is your Private Docker Registry FQDN. (<https://index.docker.io/v1/> for DockerHub)
* <your-name> is your Docker username.
* <your-pword> is your Docker password.
* <your-email> is your Docker email.

Create deployment yaml file with name ecr-deployment.yml

Ecxute the below commands

Kubectl apply –f ecr-secret.yml

Kubectl apply –f ecr-deployment.yml

Now you can DNS name in aws load balancer

We can access the application with that DNS name

http://<DNS name>:5000