In current cluster we will use some Persistent Volume provider like Heketi/GluserFS for persistence. In simplest case we can create toy PersistentVolume mounted to the host node’s filesystem directory with command:

$ mkdir /tmp/volume-local  
$ chmod -R 777 /tmp/volume-local

$ cat <<EOF | kubectl create -f -  
kind: PersistentVolume  
apiVersion: v1  
metadata:  
 name: volume-local  
 labels:  
 type: local  
spec:  
 capacity:  
 storage: 10Gi  
 accessModes:  
 - ReadWriteOnce  
 hostPath:  
 # CHANGE ME   
 path: "/tmp/volume-local"  
EOF

Сheсking if creation is successful:

$ kubectl get pv  
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS

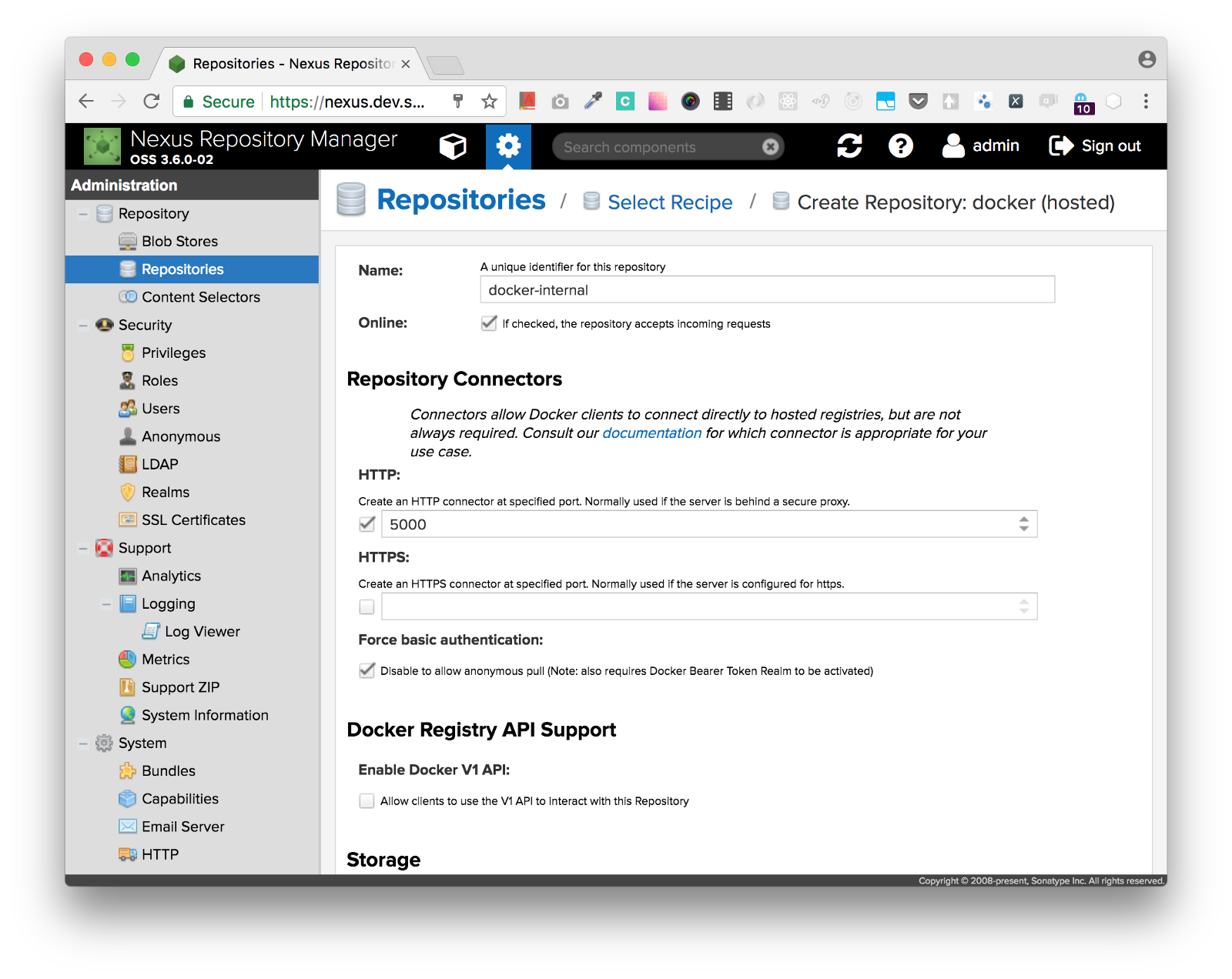
volume-local 10Gi RWO Retain Bound

CLAIM STORAGECLASS REASON AGE  
nexus/nexus-pvc 28m

Now we are ready to deploy Nexus 3. The config below creates namespace, deployment, service and ingress for Nexus 3. Do not forget to change **YOURDOMAIN.com** to your domain name.

cat <<EOF | kubectl create -f -  
apiVersion: v1  
kind: Namespace  
metadata:  
 name: nexus   
---  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
 name: nexus-pvc  
 namespace: nexus  
 labels:  
 app: nexus  
 # For GluserFS only  
 annotations:  
 volume.beta.kubernetes.io/storage-class: glusterfs-storage  
spec:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 # CHANGE ME  
 storage: 10Gi  
---  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
 name: nexus  
 namespace: nexus  
spec:  
 replicas: 1  
 template:  
 metadata:  
 labels:  
 app: nexus  
 spec:  
 containers:  
 - image: sonatype/nexus3  
 imagePullPolicy: Always  
 name: nexus  
 ports:  
 - containerPort: 8081  
 - containerPort: 5000  
 volumeMounts:  
 - mountPath: /nexus-data  
 name: nexus-data-volume  
 volumes:  
 - name: nexus-data-volume  
 persistentVolumeClaim:  
 claimName: nexus-pvc  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: nexus-service  
 namespace: nexus  
spec:  
 ports:  
 - port: 80  
 targetPort: 8081  
 protocol: TCP  
 name: http  
 - port: 5000  
 targetPort: 5000  
 protocol: TCP  
 name: docker   
 selector:  
 app: nexus  
---  
apiVersion: extensions/v1beta1  
kind: Ingress  
metadata:  
 name: nexus-ingress  
 namespace: nexus  
 annotations:  
 ingress.kubernetes.io/proxy-body-size: 100m  
 kubernetes.io/tls-acme: "true"  
 kubernetes.io/ingress.class: "nginx"  
spec:  
 tls:  
 - hosts:  
 # CHANGE ME  
 - docker.YOURDOMAIN.com  
 - nexus.YOURDOMAIN.com   
 secretName: nexus-tls  
 rules:  
 # CHANGE ME  
 - host: nexus.YOURDOMAIN.com   
 http:  
 paths:  
 - path: /  
 backend:  
 serviceName: nexus-service  
 servicePort: 80  
 # CHANGE ME  
 - host: docker.YOURDOMAIN.com   
 http:  
 paths:  
 - path: /  
 backend:  
 serviceName: nexus-service  
 servicePort: 5000  
EOF

After the deployment is done and Nexus 3 interface is available at nexus.YOURDOMAIN.com, we should create docker repository. Login to Nexus with admin/admin123, go **to “Server administration and configuration”, “Repositories”, “Create repository”, “docker (hosted)” and publish docker hosted http service on 5000** port.



Your private docker registry is ready to work at docker.YOURDOMAIN.com.

Let’s test it. Push hello-world container to registry (on your computer).

$ **docker login docker.YOURDOMAIN.com**  
User: admin  
Password: admin123  
Login Succeeded

$ docker pull dockercloud/hello-world  
$ docker tag dockercloud/hello-world docker.YOURDOMAIN.com/dockercloud/hello-world:0.1  
$ docker push docker.YOURDOMAIN.com/dockercloud/hello-world:0.1  
The push refers to a repository [docker.YOURDOMAIN.com/dockercloud/hello-world:0.1]  
b0ffedf1c11d: Pushed  
...  
8539d1fe4fab: Pushed  
latest: digest: sha256:8d9d4a28486005a6aaf1e3a16abe68b4bd82dcfe2b8602b00bee3744099fc578 size: 1570

Now test how Kubernetes pull images from our private docker registry. Create namespace for hello-world application.

cat <<EOF | kubectl create -f -  
apiVersion: v1  
kind: Namespace  
metadata:  
 name: hello-world   
EOF

Create secret in hello-world namespace.

# kubectl create secret docker-registry   
regsecret --docker-server=docker.YOURDOMAIN.com --docker-username=admin   
--docker-password=admin123 --docker-email=<your-email> --namespace   
hello-world

Deploy hello-world application and export it with NodePort. Do not forget to change YOURDOMAIN.com.

cat <<EOF | kubectl create -f -  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
 name: hello-world  
 namespace: hello-world  
spec:  
 replicas: 1  
 template:  
 metadata:  
 labels:  
 app: hello-world  
 spec:  
 containers:  
 - name: hello-world  
 image: docker.YOURDOMAIN.com/dockercloud/hello-world:0.1  
 imagePullPolicy: Always  
 ports:  
 - containerPort: 80  
 imagePullSecrets:  
 - name: regsecret  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: hello-world-service  
 namespace: hello-world  
spec:  
 ports:  
 - port: 80  
 nodePort: 30000  
 selector:  
 app: hello-world  
 type: NodePort  
EOF

Look at pod status.

$ kubectl get pods -n hello-world -o wide  
NAME READY STATUS RESTARTS AGE IP NODE  
hello-world-6f9b9bd576-b7jk6 1/1 Running 0 14s 10.233.65.102 node01

Test it.

$ curl YOUR\_CLUSTER\_PUBLIC\_IP:30000

That’s all, do not forget to change default Nexus password and disable anonymous access!