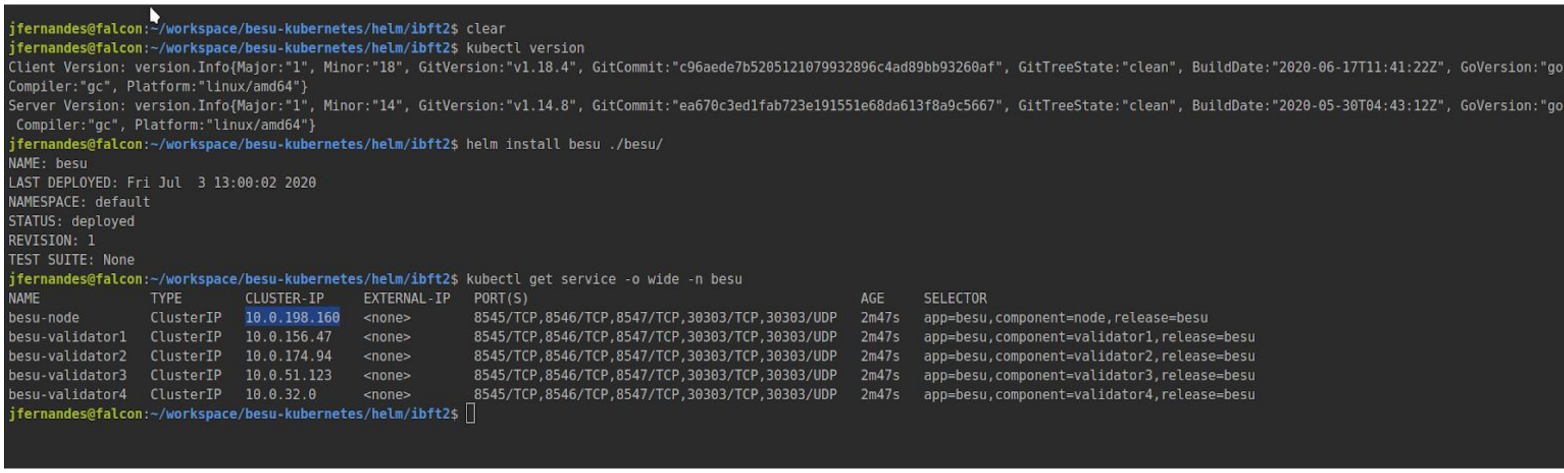
1. Deploy to Azure/GCP/AWS etc. I’ve deployed to Azure here

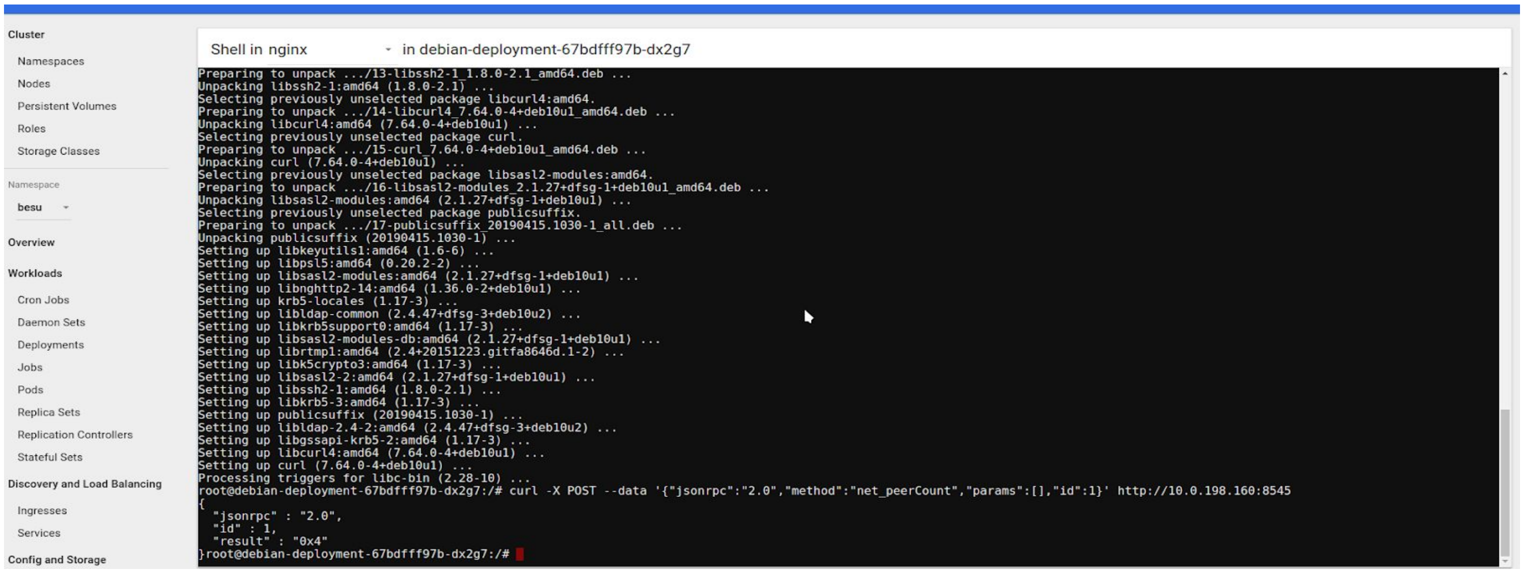


```
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ clear
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.4", GitCommit:"c96aede7b5205121079932896c4ad89bb93260af", GitTreeState:"clean", BuildDate:"2020-06-17T11:41:22Z", GoVersion:"go
Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.8", GitCommit:"ea670c3ed1fab723e191551e68da613f8a9c5667", GitTreeState:"clean", BuildDate:"2020-05-30T04:43:12Z", GoVersion:"go
 Compiler:"gc", Platform:"linux/amd64"}
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ helm install besu ./besu/
NAME: besu
LAST DEPLOYED: Fri Jul  3 13:00:02 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ kubectl get service -o wide -n besu
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                                          AGE     SELECTOR
besu-node         ClusterIP   10.0.198.160   <none>        8545/TCP,8546/TCP,8547/TCP,30303/TCP,30303/UDP   2m47s   app=besu,component=node,release=besu
besu-validator1   ClusterIP   10.0.156.47    <none>        8545/TCP,8546/TCP,8547/TCP,30303/TCP,30303/UDP   2m47s   app=besu,component=validator1,release=besu
besu-validator2   ClusterIP   10.0.174.94    <none>        8545/TCP,8546/TCP,8547/TCP,30303/TCP,30303/UDP   2m47s   app=besu,component=validator2,release=besu
besu-validator3   ClusterIP   10.0.51.123    <none>        8545/TCP,8546/TCP,8547/TCP,30303/TCP,30303/UDP   2m47s   app=besu,component=validator3,release=besu
besu-validator4   ClusterIP   10.0.32.0      <none>        8545/TCP,8546/TCP,8547/TCP,30303/TCP,30303/UDP   2m47s   app=besu,component=validator4,release=besu
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$
```

2. At this point I can connect to the besu-node service from within Azure - another container for example but not locally from my machine. Another VM connecting to the service may need an ingress depending on how the VNet and k8s cluster have been created

   I will deploy a sample debian container and attempt to connect to the besu-node service

   I’ve deployed a basic debian container which sleeps, ssh’d in and installed curl and then sent a request to the **service** ip 10.0.198.160

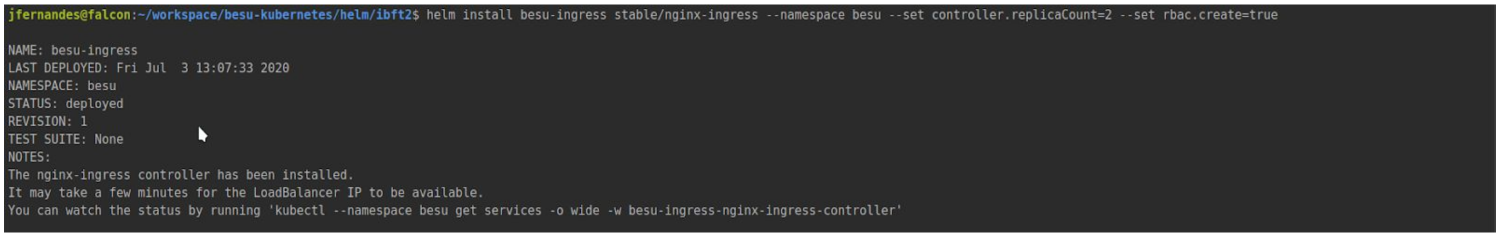


```
Preparing to unpack .../13-libssh2-1_1.8.0-2.1_amd64.deb ...
Unpacking libssh2-1:amd64 (1.8.0-2.1) ...
Selecting previously unselected package libcurl4:amd64.
Preparing to unpack .../14-libcurl4_7.64.0-4+deb10u1_amd64.deb ...
Unpacking libcurl4:amd64 (7.64.0-4+deb10u1) ...
Selecting previously unselected package curl.
Preparing to unpack .../15-curl_7.64.0-4+deb10u1_amd64.deb ...
Unpacking curl (7.64.0-4+deb10u1) ...
Selecting previously unselected package libsasl2-modules:amd64.
Preparing to unpack .../16-libsasl2-modules_2.1.27+dfsg-1+deb10u1_amd64.deb ...
Unpacking libsasl2-modules:amd64 (2.1.27+dfsg-1+deb10u1) ...
Selecting previously unselected package publicsuffix.
Preparing to unpack .../17-publicsuffix_20190415.1030-1_all.deb ...
Unpacking publicsuffix (20190415.1030-1) ...
Setting up libkeyutils1:amd64 (1.6-6) ...
Setting up libpsl5:amd64 (0.20.2-2) ...
Setting up libsasl2-modules:amd64 (2.1.27+dfsg-1+deb10u1) ...
Setting up libnghttp2-14:amd64 (1.36.0-2+deb10u1) ...
Setting up krb5-locales (1.17-3) ...
Setting up libldap-common (2.4.47+dfsg-3+deb10u2) ...
Setting up libkrb5support0:amd64 (1.17-3) ...
Setting up libsasl2-modules-db:amd64 (2.1.27+dfsg-1+deb10u1) ...
Setting up librtmp1:amd64 (2.4+20151223.gitfa8646d.1-2) ...
Setting up libk5crypto3:amd64 (1.17-3) ...
Setting up libsasl2-2:amd64 (2.1.27+dfsg-1+deb10u1) ...
Setting up libssh2-1:amd64 (1.8.0-2.1) ...
Setting up libkrb5-3:amd64 (1.17-3) ...
Setting up publicsuffix (20190415.1030-1) ...
Setting up libldap-2.4-2:amd64 (2.4.47+dfsg-3+deb10u2) ...
Setting up libgssapi-krb5-2:amd64 (1.17-3) ...
Setting up libcurl4:amd64 (7.64.0-4+deb10u1) ...
Setting up curl (7.64.0-4+deb10u1) ...
Processing triggers for libc-bin (2.28-10) ...
root@debian-deployment-67bdfff97b-dx2g7:/# curl -X POST --data '{"jsonrpc":"2.0","method":"net_peerCount","params":[],"id":1}' http://10.0.198.160:8545
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "0x4"
}root@debian-deployment-67bdfff97b-dx2g7:/#
```

Any container in the cluster should be able to talk to the besu-node service directly without issue to deploy contracts etc
Alternatively, to connect from my local machine to the Azure besu-node service

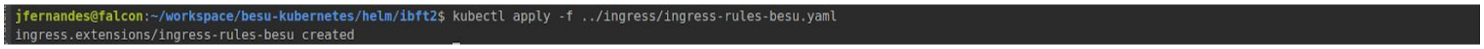
3. To connect to the besu-node service from my local machine, I deploy an ingress which routes me to the pod behind the service. Please note that all comms is to the **service** and not the pod

   Note: I'm deploying this directly from the samples, when doing this please update the config with your certs and then deploy



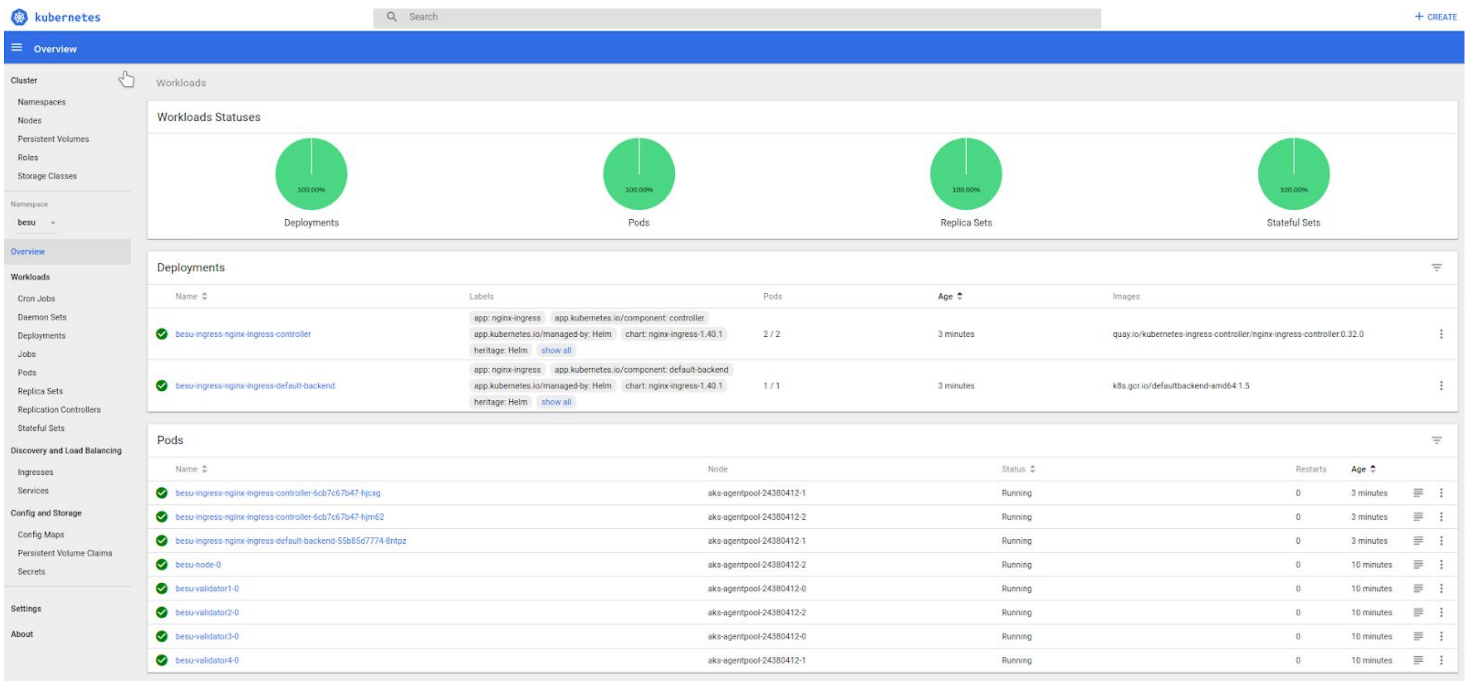
```
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ helm install besu-ingress stable/nginx-ingress --namespace besu --set controller.replicaCount=2 --set rbac.create=true

NAME: besu-ingress
LAST DEPLOYED: Fri Jul  3 13:07:33 2020
NAMESPACE: besu
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The nginx-ingress controller has been installed.
It may take a few minutes for the LoadBalancer IP to be available.
You can watch the status by running 'kubectl --namespace besu get services -o wide -w besu-ingress-nginx-ingress-controller'
```

Then deploy ingress rules to route back to that node



```
jfernandes@falcon:~/workspace/besu-kubernetes/helm/ibft2$ kubectl apply -f ../ingress/ingress-rules-besu.yaml
ingress.extensions/ingress-rules-besu created
```

This is what that looks like on AKS



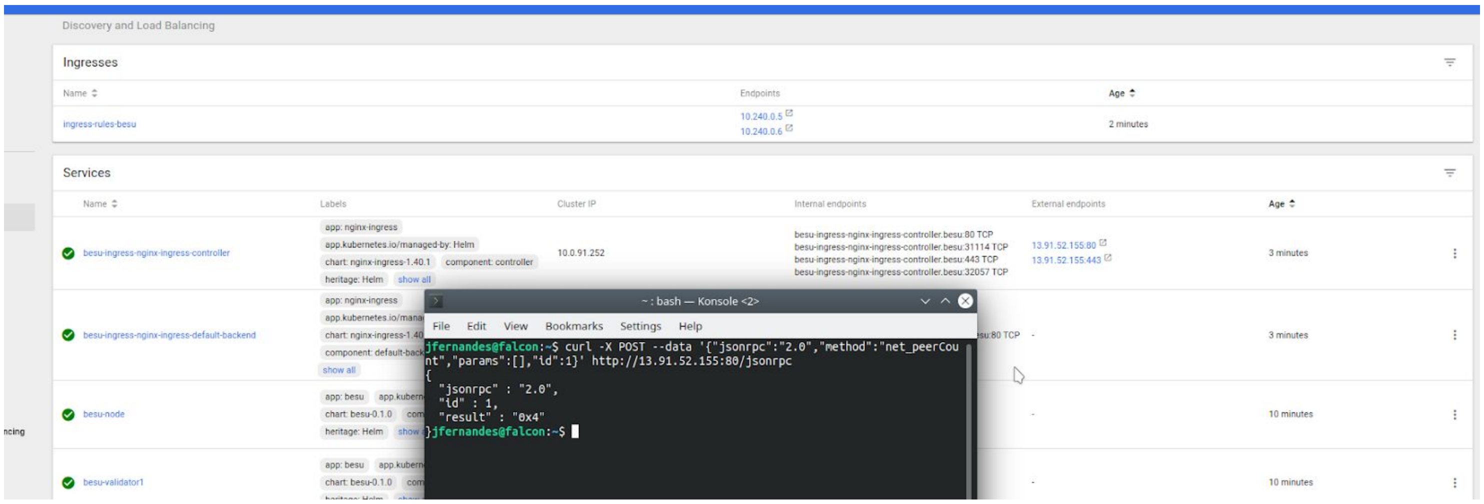
Deployments

Name	Labels	Pods	Age	Images
besu-ingress-nginx-ingress-controller	app: nginx-ingress app.kubernetes.io/component: controller app.kubernetes.io/managed-by: Helm chart: nginx-ingress-1.40.1 heritage: Helm show all	2 / 2	3 minutes	quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.32.0
besu-ingress-nginx-ingress-default-backend	app: nginx-ingress app.kubernetes.io/component: default-backend app.kubernetes.io/managed-by: Helm chart: nginx-ingress-1.40.1 heritage: Helm show all	1 / 1	3 minutes	k8s.gcr.io/defaultbackend-amd64:1.5

Pods

Name	Node	Status	Restarts	Age
besu-ingress-nginx-ingress-controller-6cb7c67b47-hjcxg	aks-agentpool-24380412-1	Running	0	3 minutes
besu-ingress-nginx-ingress-controller-6cb7c67b47-hjm62	aks-agentpool-24380412-2	Running	0	3 minutes
besu-ingress-nginx-ingress-default-backend-55b85d7774-8ntpz	aks-agentpool-24380412-1	Running	0	3 minutes
besu-node-0	aks-agentpool-24380412-2	Running	0	10 minutes
besu-validator1-0	aks-agentpool-24380412-0	Running	0	10 minutes
besu-validator2-0	aks-agentpool-24380412-2	Running	0	10 minutes
besu-validator3-0	aks-agentpool-24380412-0	Running	0	10 minutes
besu-validator4-0	aks-agentpool-24380412-1	Running	0	10 minutes

Now I can connect to the besu-node service from my local machine via the Ingress IP. Note the extra `/jsonrpc` path I've added as per the ingress rules deployed.



Ingresses

Name	Endpoints	Age
ingress-rules-besu	10.240.0.5 10.240.0.6	2 minutes

Services

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
besu-ingress-nginx-ingress-controller	app: nginx-ingress app.kubernetes.io/managed-by: Helm chart: nginx-ingress-1.40.1 component: controller heritage: Helm show all	10.0.91.252	besu-ingress-nginx-ingress-controller.besu:80 TCP besu-ingress-nginx-ingress-controller.besu:31114 TCP besu-ingress-nginx-ingress-controller.besu:443 TCP besu-ingress-nginx-ingress-controller.besu:32057 TCP	13.91.52.155:80 13.91.52.155:443	3 minutes
besu-ingress-nginx-ingress-default-backend	app: nginx-ingress app.kubernetes.io/mana chart: nginx-ingress-1.40 component: default-back show all		su:80 TCP	-	3 minutes
besu-node	app: besu app.kubern chart: besu-0.1.0 com heritage: Helm show			-	10 minutes
besu-validator1	app: besu app.kubern chart: besu-0.1.0 com			-	10 minutes

```
jfernandes@falcon:~$ curl -X POST --data '{"jsonrpc":"2.0","method":"net_peerCou
nt","params":[],"id":1}' http://13.91.52.155:80/jsonrpc
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "0x4"
}jfernandes@falcon:~$
```