ATOM Software Deployment Guide

version 11.8

Table of Contents

Purpose of this document

This document is intended for deploying ATOM software in a Kubernetes environment.

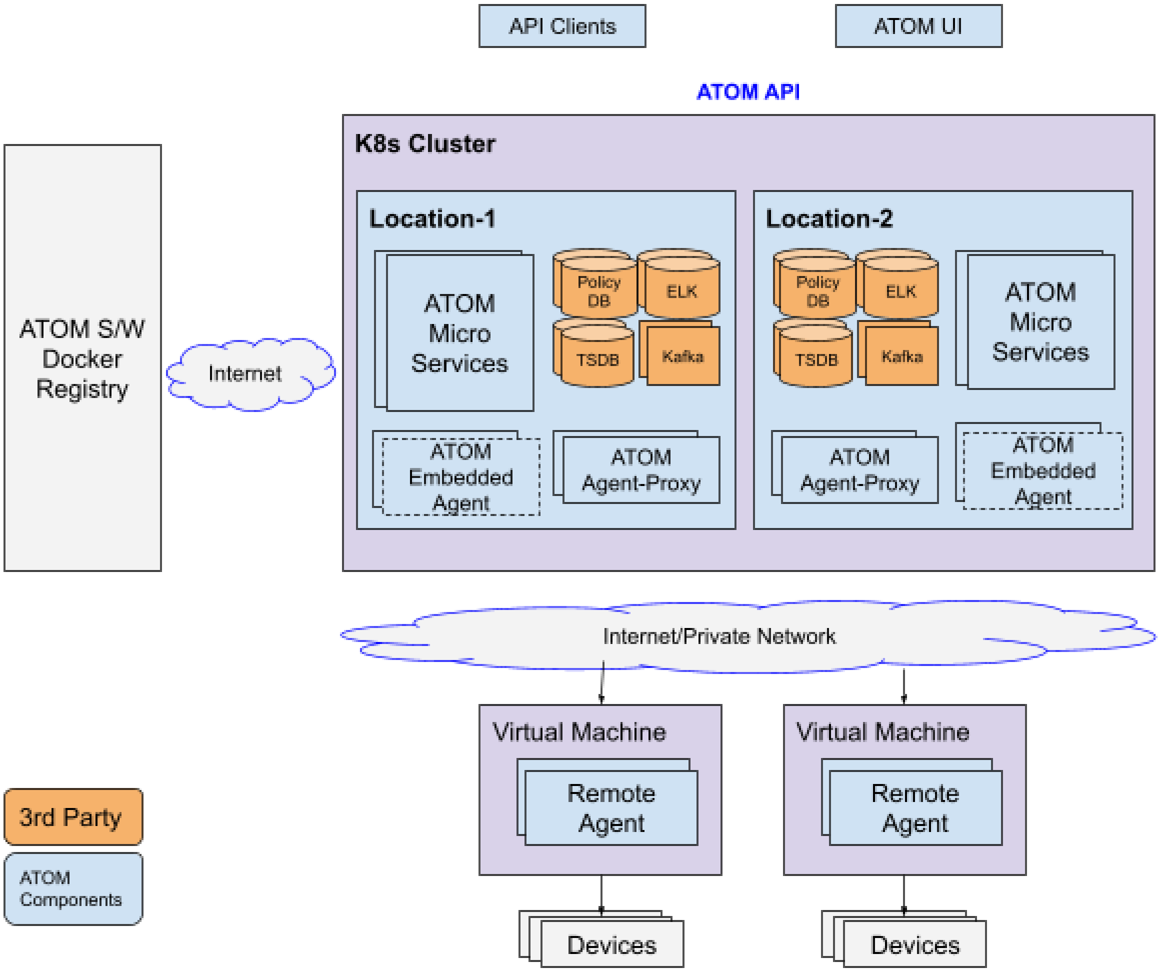
Intended Audience

The procedure for installing the ATOM software is meant for administration teams responsible for ATOM software deployment and operations.

ATOM deployment and operations requires hands-on experience installing Kubernetes clusters and deployment using Helm charts. This document assumes that you are familiar with Docker, containers, hypervisors, networking, and a good working knowledge of the operating systems.

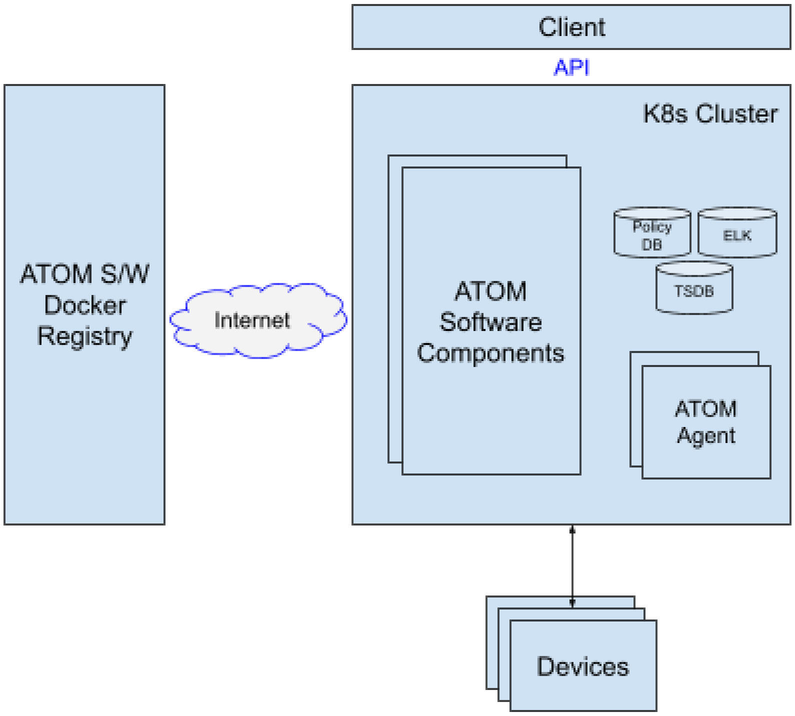
Overview of ATOM Architecture

ATOM software is containerized and runs on a Kubernetes cluster. ATOM is provided as a self-contained installation package with all the required components:



ATOM Deployment

ATOM deployment requires software components to be deployed in a Kubernetes environment. Software will be distributed through a central repository.



Deployment scenarios

ATOM can be deployed in any of the following environments:

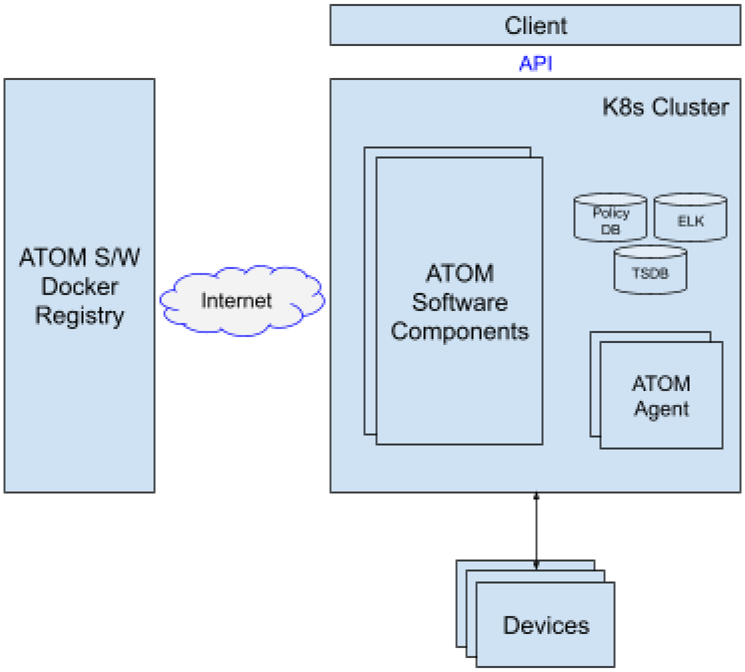
- On-Prem Kubernetes
- Google Cloud Platform (GCP)
- Amazon web service (AWS)

ATOM can be deployed with all the components at a single location or some of the components distributed.

- Local Deployment
- Distributed Deployment

Local Deployment

Local deployment has all the ATOM software components deployed in a single Kubernetes cluster.

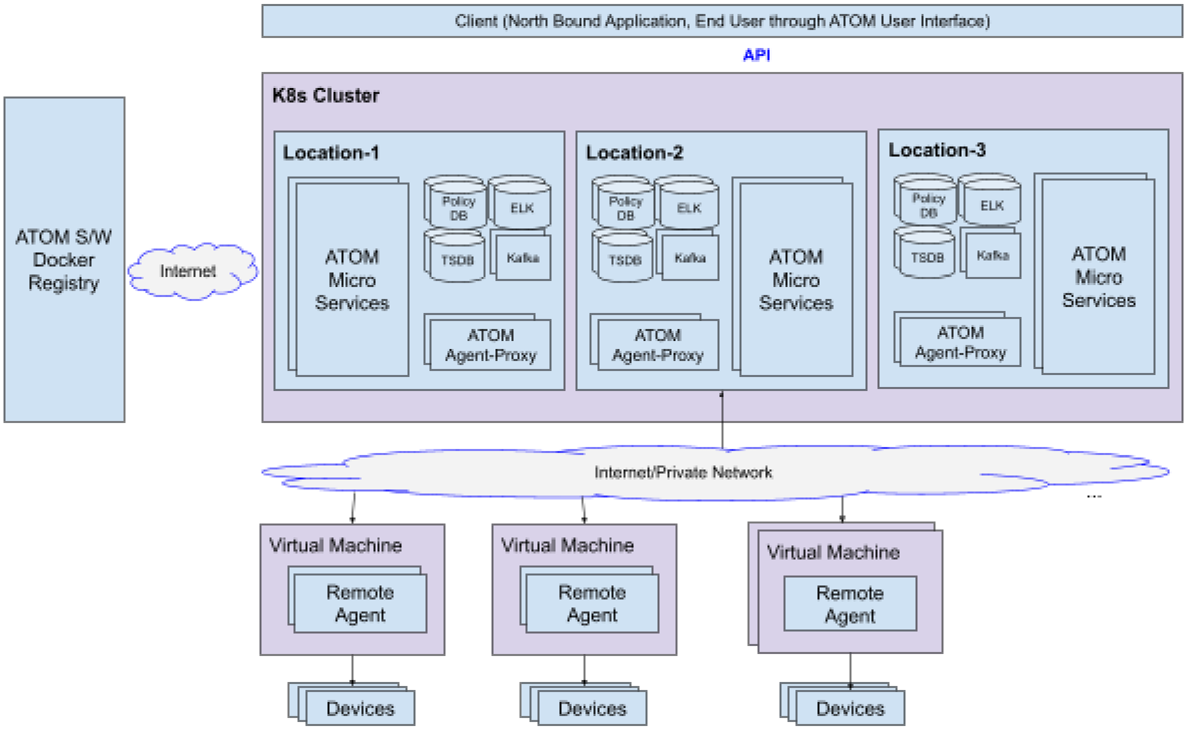


Distributed Deployment

Distributed deployment allows ATOM software components to be distributed across multiple logical or geographical locations. Distributed Deployment is applicable in the following scenarios:

1. Remote Agent - In some scenarios network equipment is distributed across different locations. ATOM Agent can be deployed close to the Network equipment for Security or performance reasons.

2. Geo-redundant HA - ATOM Components can be deployed across multiple Locations/Sites within the same region to provide Fault Tolerance against an entire Site/Location going down. More details in ATOM Multi Availability Zone based HA.



Target Infrastructure

ATOM Can be deployed On Premises, Cloud or a combination as summarized in the Table below.

Environment	Description	Use case	Prerequisites
Cloud (Amazon, GCP or Similar)	Typically for Staging & Production Deployments	Development Stage Production	Hardware Requirements
On Premises	Typically for Staging & Production Deployments Can be used for Multi user shared Development as well	Development Stage Production	● On-Prem VMware ESXi, KVM ● Hardware Requirements
Cloud + On Premise	ATOM Agent can be deployed on-Premises while rest of ATOM can be deployed in the Cloud	Development Stage Production	● On-Prem VMware ESXi, KVM ● Hardware Requirements

On-Prem VMware ESXi, KVM

For the Kubernetes cluster deployed on ESXi, KVM etc., make sure required Compute, Storage & Memory resources for VM nodes are allocated to have ATOM running on top of K8s cluster.

Anuta provides the OVA images for K8s Master and Worker nodes creation on ESXi, while the OVA’s can be converted to Qcow2 images to deploy K8s Master and Worker nodes on KVM.

Cloud (GCP / AWS)

As cloud deployments on GCP/AWS offer different variants of node-types, make sure the Node Type you selected matches the resources required for a Worker Node mentioned in Compute, Storage & Memory requirements(Separate Master Node not required in GCP/AWS).
For GCP deployment a *e2-highmem-4* or *custom-4-32768-ext* Node type would be required and a r6i.xlarge Node type for AWS deployment.

Requirements

Before deploying ATOM in the kubernetes cluster, ensure that the following requirements are satisfied:

1. Hardware Requirements
2. Network Requirements
3. Kubernetes Cluster Requirements
4. Software Requirements

Compute, Storage & Memory

Note:

> SSD storage is **mandatory** as the ATOM’s databases and messaging services will perform better over SSDs. When using local storage, it is recommended to use **RAID10** based **Storage** and provision VMs across multiple physical servers.

Overview

Type	**Minimal 1 DC**	**Resilient 1 DC**	**Resilient 2 DC**	**Resilient 3 DC**
Master Nodes Per DC	3	3	2 and 1*	1
Master Specs	4 vCPU, 32GB RAM, 300GB SSD	4 vCPU, 8GB RAM, 50GB SSD	4 vCPU, 8GB RAM, 50GB SSD	4 vCPU, 8GB RAM, 50GB SSD
Master Node shares workload	Yes	No	No	No
Worker Nodes Per DC	1	9	4	3

Worker Specs	4 vCPU, 32GB RAM, 300GB SSD	4 vCPU, 32GB RAM, 300GB SSD	4 vCPU, 32GB RAM, 300GB SSD	4 vCPU, 32GB RAM, 300GB SSD
Total Nodes	4	12	11	12
Total IP requirements	4+3(VIP)=7	12+3(VIP)=15	11+3(VIP)=14	12+3(VIP)=15

Refer IP Addressing Section for more details
*Refer 2 Sites Deployment section below

Minimal Setup

A Minimal setup that doesn't support resiliency for ATOM components but has Kubernetes HA needs a cluster (3 masters and 1 worker node) based out of ESXi with recommendations listed below

Component	Requirements Description
K8s Master - 3 nodes	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB
K8s Workers - 1 node	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB

Total IP Address: 4 IPs + 3 VIPs = 7 IPs. Refer IP Addressing Section for more details.
In this model K8s Master shares the workloads of ATOM components as well.

Resilient HA Setup

HA setup supporting resiliency with regards to one node or pod failures requires a Kubernetes cluster (3 masters and 9 worker nodes) based out of ESXi with the following details

Component	Requirements Description
K8s Master - 3 nodes	For each node storage reserved in ESXi = 50 GB (SSD) ● CPU - 4 vCPU ● Memory - 8GB
K8s Workers - 9 nodes	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB

Total IP Address: 12 IPs + 3 VIPs = 15 IPs. Refer IP Addressing Section for more details.
In this model K8s Master shares the workloads of ATOM components as well.

Multi-site Deployment (Remote ATOM agent)

For a Multi-site distributed deployment, where the ATOM agent is deployed remotely, a single ATOM agent (minimum) is deployed at each site in addition to the above setup choices. A Virtual Machine with below minimum spec is required at each site location(s):

Component	Requirements Description
1 Virtual Machine	Storage reserved in ESXi = 50 GB (SSD) ● CPU - 4 vCPU ● Memory - 8GB

Total IP Address: 1 IP. Refer IP Addressing Section for more details

ATOM Multi Availability Zone HA Deployment

ATOM supports deployment across multiple sites (aka Availability Zones) to support high availability in the event of a site failure provided these sites are connected over low latency links. This requires ATOM Components to be deployed across multiple sites or Availability Zones (AZs). Availability Zones are available when workloads are provisioned in a Cloud Service Provider. In this scenario, Kubernetes Cluster extends to multiple sites/Zones.

If ATOM is deployed in a single location, it is recommended that master nodes and worker nodes are at least deployed on 3 separate physical servers. In such cases, ATOM will continue to be available in the event of a single physical machine failure.

References:

- https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/RegionsAndAZs.html
- https://docs.microsoft.com/en-us/azure/availability-zones/az-overview
- https://cloud.google.com/compute/docs/regions-zones

Caveats:

1. Full Fault Tolerance against one Site failure requires ATOM deployment across 3 Locations/Sites.
2. In case only 2 Sites/Locations are available:
    a. Full Fault Tolerance against one Site failure is supported, however, due to the quorum requirements of some of the components like etcd, manual intervention may be needed if the site that has majority is down.
3. Multi-region deployment is not supported, ATOM clusters can only be deployed across multiple AZs within a region due to low latency requirements (<10 ms).
4. Some ATOM Components that support deployment across multiple Availability Zones or sites are sensitive to Latency. In such scenarios, there will be an impact on application performance or throughput

3 Sites Deployment:

For Each Site:

Component	Requirements Description
K8s Master - 1 nodes	For each node storage reserved in ESXi = 50 GB (SSD) ● CPU - 4 vCPU ● Memory - 8GB
K8s Workers - 3 nodes	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB

Total IP Address across 3 sites: 12 IPs + 3 VIPs = 15 IPs. Refer IP Addressing Section for more details

2 Sites Deployment:

Site-1:

Component	Requirements Description
K8s Master - 2 nodes	For each node storage reserved in ESXi = 50 GB (SSD) ● CPU - 4 vCPU ● Memory - 8GB
K8s Workers - 4 nodes	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB

Site-2:

Component	Requirements Description
K8s Master - 1* nodes	For each node storage reserved in ESXi = 50 GB (SSD) ● CPU - 4 vCPU ● Memory - 8GB
K8s Workers - 4 nodes	For each node storage reserved in ESXi = 300 GB (SSD) ● CPU - 4 vCPU ● Memory - 32GB

Total IP Address across 2 sites: 11 IPs + 3 VIPs = 14 IPs. Refer IP Addressing Section for more details

*In site-1 disaster case site-2 needs two master nodes to be deployed instead of one. Hence the required second master spec needs to be kept available ahead to handle disaster of site-1.

AWS Availability Zones

Refer to section Deploying New K8s Cluster for ATOM deployment in AWS which uses the Availability Zones(AZ) during deployment.

On Premises Across Data Centers / Locations

For on-premises deployment of a Multi Availability Zone Model across different sites, latency requirements have to be met.

Refer to section Deploying New Kubernetes Cluster for On Premises ATOM deployment which creates K8s cluster among Master and Worker Nodes across the ESXis/Locations/DataCenters having reachability.

Network Requirements

ATOM related Ports/Protocols

Each of the components of the ATOM application communicate with each other and external using the following ports and protocols.

Wherever applicable, Firewall rules need to be updated to allow the communication between external clients to ATOM or from ATOM software to network infrastructure or between ATOM software components.

End Points	Port	Communication protocol	Notes
Northbound communication [External clients, access to ATOM Portal, and other ATOM Mgmt Clients]			
On-Prem Deployment of ATOM			
ATOM Server (End user Application-NonSSO)	30443	HTTPS access	This will be the ATOM GUI page served via HAproxy.
Single Sign-On	443	HTTPS access	For Single Sign-On login over the VIP to access ATOM, Grafana, Kibana, Glowroot and Kafka-Manager
Minio	31311	HTTP access	ATOM FileServer/Minio Access
AWS Deployment of ATOM			
Single Sign-On	443	HTTPS access	For Single Sign-On login over the VIP to access ATOM, Grafana, Kibana, Glowroot and Kafka-Manager
Minio	31311(NodePort)	HTTP access	ATOM FileServer/Minio Access

Inter-components communication [Applicable when ATOM agent and server components are deployed separately with possibly a firewall in between]			
ATOM Server - ATOM Agent	7000	TCP/RSocket	Remote Agent communicates with Agent-proxy via agent-lb.
ATOM Agent - Application Performance Monitoring	8181	TCP/Grpc	Remote Agent drops performance metrics to central APM
Southbound communication with network devices from ATOM Agent [Applicable when a Firewall is present between ATOM agent and devices/NetworkElements]			
ATOM Agent - Network Elements	80	Hypertext transfer protocol (HTTP) (IN/OUT)	OUT - Outbound IN - Inbound Different ports are used for various use cases in ATOM. Make sure PING reachability is also there.
	443	Hypertext transfer protocol secure (HTTPS) (IN/OUT)	
	23	Telnet to network devices (TCP) (OUT)	
	21	FTP to network device (TCP) (IN)	
	22	SSH to network devices (TCP) (OUT)	
	161	SNMP to network devices (UDP) (OUT)	
	162	SNMP Trap Listening (Server) from network devices (UDP) (IN)	
	69	TFTP to network devices (UDP) (IN)	
	514	SYSLOG Listening (Server) port from network devices (UDP) (IN)	
	830	NETCONF to network devices(TCP) (OUT)	
	12455	Telemetry Server for TCP Communication (IN)	
	12456	Telemetry Server for UDP Communication (IN)	
	12454	Telemetry GRPC Server (IN)	
	2055	Netflow UDP Server (IN)	

Please ensure that public access is available on all the nodes. If public access cannot be provided across the nodes then we need to consider an offline mode of installation of Atom Software by hosting Registry within your network.

Below are details of public domains which ATOM would access for pulling docker images and other binaries.

	Port/Protocol	Domain Name
ATOM -> Required Public Access Details for Firewall if applicable.	443/https	registry-1.docker.io
	443/https	quay.io
	443/https	gcr.io
	443/https	grafana.com
	443/https	codeload.github.com
	443/https	deb.debian.org
	443/https	registry.opensource.zalan.do
	443/https	ghcr.io

Kubernetes related Ports/Protocols(on-prem)

Below are Ports and Protocols which need to be allowed for Kubernetes cluster creation among VM nodes. These need to be allowed in Firewall if in between VMs there is a Firewall when VMs are spread across DCs etc..

Ports	Protocol	Notes
443	TCP	kubernetes API server(HA mode)
6443	TCP	kubernetes API server
2379-2380	TCP	etcd server client API
10250	TCP	Kubelet API
10251	TCP	Kube-scheduler
10252	TCP	Kube-controller-manager
10255	TCP	Kubelet
179	TCP	Calico CNI
9100	TCP	Prometheus
30000-32767	TCP	NodePort services

6783	TCP	Weaveport(deprecated)

Linstor related Ports/Protocols(on-prem)

ATOM uses linstor CSI driver as a storage provisioner on on-premises deployments. Below specified Ports and Protocols need to be allowed for Kubernetes among VM nodes related to Linstor. If the kubernetes cluster spreads across multiple DCs, these ports and protocols need to be open on the DC firewalls as well.

Protocol: TCP, Ports: 3366-3367, 3370, 3376-3377, 7000-8000

IP Addressing Requirements

- One IP for each of the VM nodes.
- For Minimal and HA Master setup, when 3 Masters are used, reserve one extra IP(virtual IP) belonging to the same subnet as other 3 Masters.
- Two IPs(virtual IP) for application internal load-balancing related to api-server & agents.
- IP addresses of all kubernetes nodes and virtual IPs should be in the same L2 segment.
- Subnet **10.200.0.0/16** subnet is used internally in the ATOM kubernetes cluster for communication between microservices. If this subnet conflicts with any of the existing network device IPs then a different subnet of size /16 shall be chosen. Update the file wrapper.properties with the chosen subnet.

Kubernetes Cluster Requirements

ATOM Software needs to be installed on a dedicated kubernetes cluster and it can be deployed on the following Kubernetes Distributions:

1. Amazon EKS
2. Google GKE
3. Upstream Kubernetes (https://github.com/kubernetes/kubernetes). Anuta provides CentOS based OVAs/QCOW2 images. These are customized images that include all the required software components. ATOM kubernetes cluster can only be installed on nodes created by these images.

Any other Kubernetes Distribution or node OS distribution requires additional validation from Anuta and requires significant lead time depending on the distribution.

Anuta provides deployment artifacts such as OVA/QCOW2 images for master and worker nodes, scripts for creating the Kubernetes cluster and the container images required for deploying ATOM.

For creating a Kubernetes cluster, check if the following requirements are satisfied:

1. All the hardware requirements defined in the section, Hardware Requirements are met.

2. Anuta provided OVAs (Centos server with pre-installed minimal packages) are already imported into the vCenter template library.
3. Use Anuta provided QCOW2 images for deploying on KVM.
4. Static IPs are assigned to master and worker nodes

For bootstrapping the Kubernetes cluster, run the installation script. Installation script will need inputs like VM IPs, Gateway, Netmask, DNS Server, NTP Server details and will install all the required components such as

- Docker-ce
- Kubectl
- Helm

Once the Kubernetes cluster is formed, the ATOM deployment can be done subsequently. Refer to the section, “Procedure for Deploying ATOM”.

Deployment scripts and files

To simplify the deployment of Kubernetes clusters in your environment, the required scripts and files are organized into folders and are provided by Anuta Networks (in a zipped format).

Name of the file/folder	Description
ATOM	ATOM’s deployment files
node_setup.py	Helper Script to bootstrap the nodes and install the atom software.

ATOM Software Requirements

Before proceeding with the deployment of ATOM application, you must have the following software artifacts with you, obtained from Anuta Networks:

- Deployment Images
- Deployment scripts and files

Deployment Images

All the images required for deploying the components of ATOM will be pulled from the repositories, created in Quay (https://quay.io/repository/).

The images have been tagged with a specific name, in the format given below:

```
quay.io/<organization>/<image name>:<tag>
Example: quay.io/release/atom-core:11.X.X.X.YYYYY
```

Deployment scripts and files

Deploying ATOM in the local setup involves deploying the components required to build the ATOM application using Helm charts. To simplify the deployment in your environment, the required scripts and files are organized into folders and are provided by Anuta (in a zipped format).

Name of the file/folder	Description
ATOM	ATOM's deployment files
scripts	Check and install kubernetes, docker, helm, python packages

The key folder **ATOM**, contains Helm charts, templates and the deployment scripts which will be used for ATOM deployment. It has Helm charts like below

- **databases** -- contains the deployment files of all databases - PolicyDB and kafka
- **atom** -- contains multiple charts of individual microservice
- **Infra** -- contains charts related to infra components such as web-proxy, logstash, glowroot etc.
- **external-services** -- optional services to access external services like databases, kafka etc.
- **grafana** -- contains the helm charts for Grafana monitoring tool
- **persistence** -- contains the yaml files for creating persistent volumes
- **tsdb-server and and tsdb-monitoring** -- contains the helm charts for tsdb
- **minio** -- contains helm charts for minio/object storage
- **sso** -- contains helm charts for sso objects
- **metallb -** contains helm charts for providing virtual load balancing service

Each of the above folders contains the following:

1. *README.md* - Detailed readme information
2. *chart.yaml* - Contains the information about the chart
3. *values.yaml* - Default configuration values for this chart
4. templates - A directory of templates containing the template, which when combined with values provided in the run-time generate a valid Kubernetes manifest file.

Security Apps on VM nodes before ATOM install

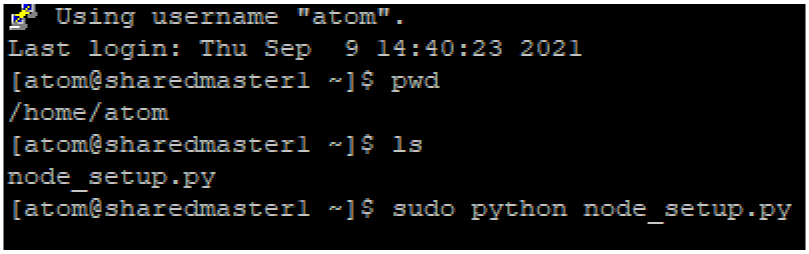
Users can install any security agents or clients on the VM nodes to meet their internal security compliance policies. Example - Trend Micro. Users have to make sure that these agents or clients shall not interfere with kubernetes processes and applications so that they are not modified when the ATOM is in running state. For information on ports that are used by Kubernetes and ATOM applications, please refer to section Networking Requirements.

Procedure for Deploying ATOM on-prem

ATOM applications can be deployed on new Kubernetes with help of Deployment scripts and files provided by Anuta.

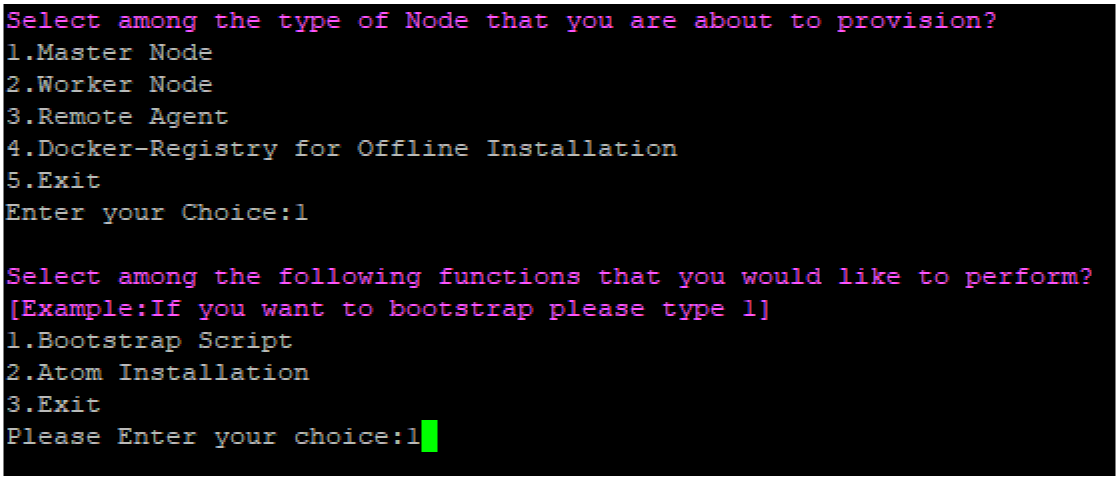
New Kubernetes cluster

1. Verify that you have imported the shared Anuta OVA templates into your VMware vCenter.
2. Create master nodes and worker nodes - See section requirements for more details
3. Login credentials for these nodes will be **atom/secret@123**. For any python script executions use **sudo** for which password is again secret@123
   NOTE: Do not login with **root** username into VMs
4. Run the node_setup.py which is present in the home directory using sudo privileges as shown below [Note:This script needs to be run on each node individually]:



```
 Using username "atom".
Last login: Thu Sep  9 14:40:23 2021
[atom@sharedmaster1 ~]$ pwd
/home/atom
[atom@sharedmaster1 ~]$ ls
node_setup.py
[atom@sharedmaster1 ~]$ sudo python node_setup.py
```

5. Enter 1 (master) or 2(worker) depending on the type of node that you want to provision.

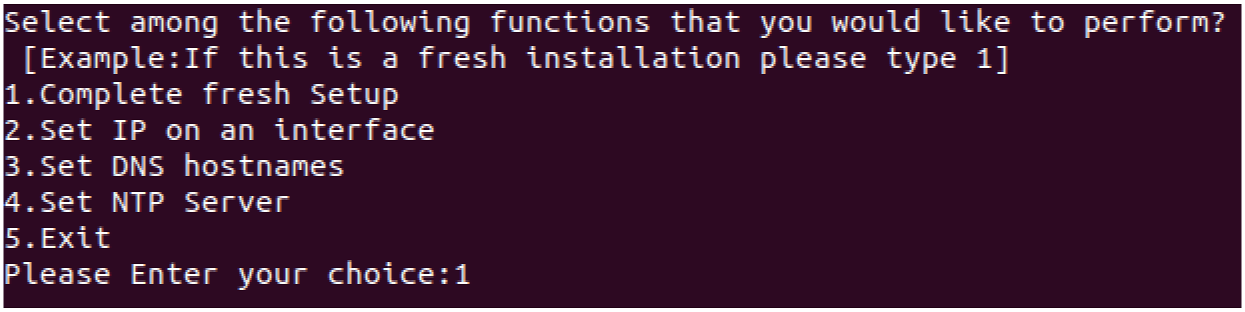


```
Select among the type of Node that you are about to provision?
1.Master Node
2.Worker Node
3.Remote Agent
4.Docker-Registry for Offline Installation
5.Exit
Enter your Choice:1

Select among the following functions that you would like to perform?
[Example:If you want to bootstrap please type 1]
1.Bootstrap Script
2.Atom Installation
3.Exit
Please Enter your choice:1
```

Choose among the following:

   1. Bootstrap Script: This script will initially help you set up basic Network Connectivity, Hostname configuration and NTP settings.
   2. Atom Installation: This script will be used to deploy k8s and bring up the atom software at a later stage. Complete steps 4-7 before invoking this.

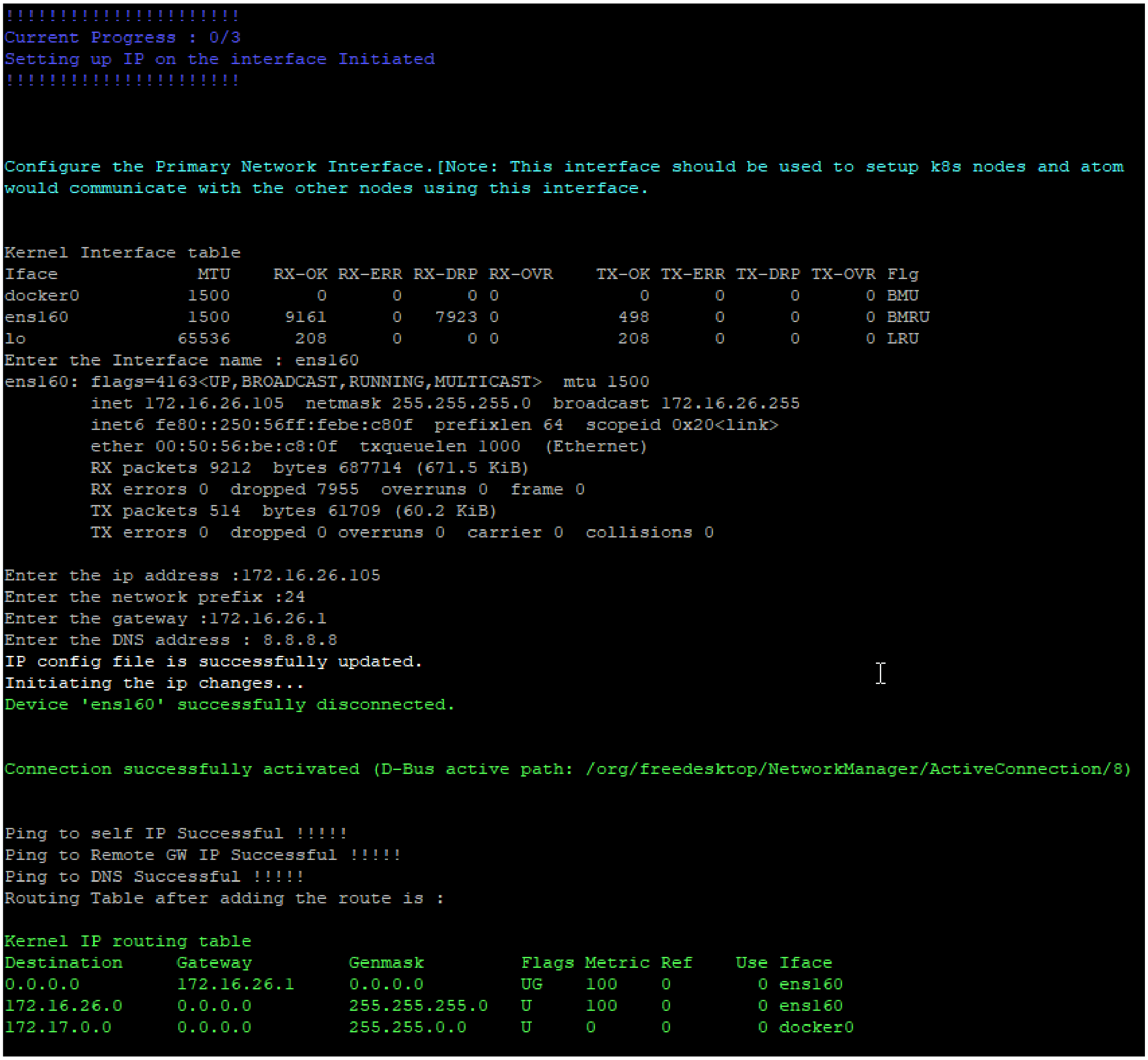
6. Enter 1 to proceed with the bootstrap function and select the complete fresh setup by again choosing 1 as shown below:



```
Select among the following functions that you would like to perform?
 [Example:If this is a fresh installation please type 1]
1.Complete fresh Setup
2.Set IP on an interface
3.Set DNS hostnames
4.Set NTP Server
5.Exit
Please Enter your choice:1
```

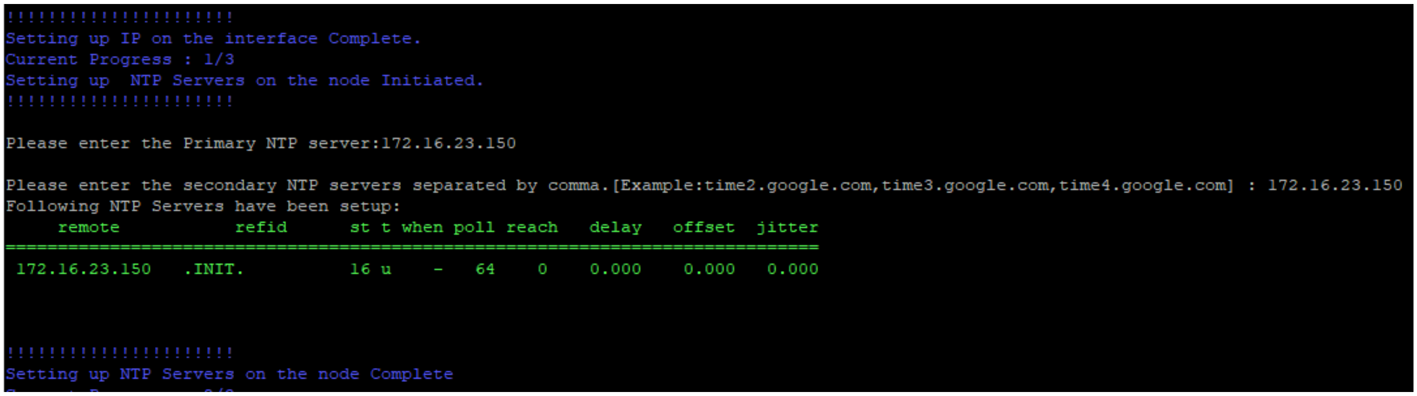
7. Provide the following inputs as requested by the script:
   1. Interface Details to be provisioned along with relevant CIDR info.
   2. DNS Server Information
   3. NTP Server Information
   4. Hostname of the VM along with the hostname-ip to bind.

   Refer the screenshot below:



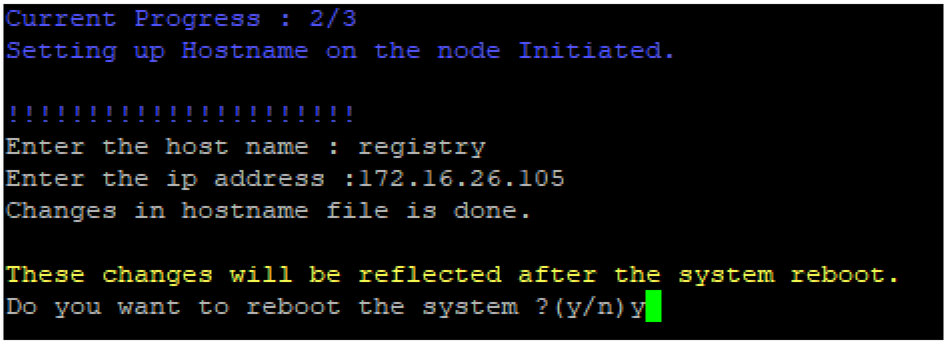
```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 0/3
Setting up IP on the interface Initiated
!!!!!!!!!!!!!!!!!!!!!!



Configure the Primary Network Interface.[Note: This interface should be used to setup k8s nodes and atom
would communicate with the other nodes using this interface.


Kernel Interface table
Iface             MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
docker0          1500        0      0      0 0             0      0      0      0 BMU
ens160           1500     9161      0   7923 0           498      0      0      0 BMRU
lo              65536      208      0      0 0           208      0      0      0 LRU
Enter the Interface name : ens160
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.26.105  netmask 255.255.255.0  broadcast 172.16.26.255
        inet6 fe80::250:56ff:febe:c80f  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:be:c8:0f  txqueuelen 1000  (Ethernet)
        RX packets 9212  bytes 687714 (671.5 KiB)
        RX errors 0  dropped 7955  overruns 0  frame 0
        TX packets 514  bytes 61709 (60.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Enter the ip address :172.16.26.105
Enter the network prefix :24
Enter the gateway :172.16.26.1
Enter the DNS address : 8.8.8.8
IP config file is successfully updated.
Initiating the ip changes...
Device 'ens160' successfully disconnected.


Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/8)


Ping to self IP Successful !!!!!
Ping to Remote GW IP Successful !!!!!
Ping to DNS Successful !!!!!
Routing Table after adding the route is :

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.26.1     0.0.0.0         UG    100    0        0 ens160
172.16.26.0     0.0.0.0         255.255.255.0   U     100    0        0 ens160
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
```

Network Configuration Details



```
!!!!!!!!!!!!!!!!!!!!!!
Setting up IP on the interface Complete.
Current Progress : 1/3
Setting up  NTP Servers on the node Initiated.
!!!!!!!!!!!!!!!!!!!!!!

Please enter the Primary NTP server:172.16.23.150

Please enter the secondary NTP servers separated by comma.[Example:time2.google.com,time3.google.com,time4.google.com] : 172.16.23.150
Following NTP Servers have been setup:
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 172.16.23.150   .INIT.          16 u    -   64    0    0.000    0.000   0.000



!!!!!!!!!!!!!!!!!!!!!!
Setting up NTP Servers on the node Complete
```

NTP Server Configuration Details

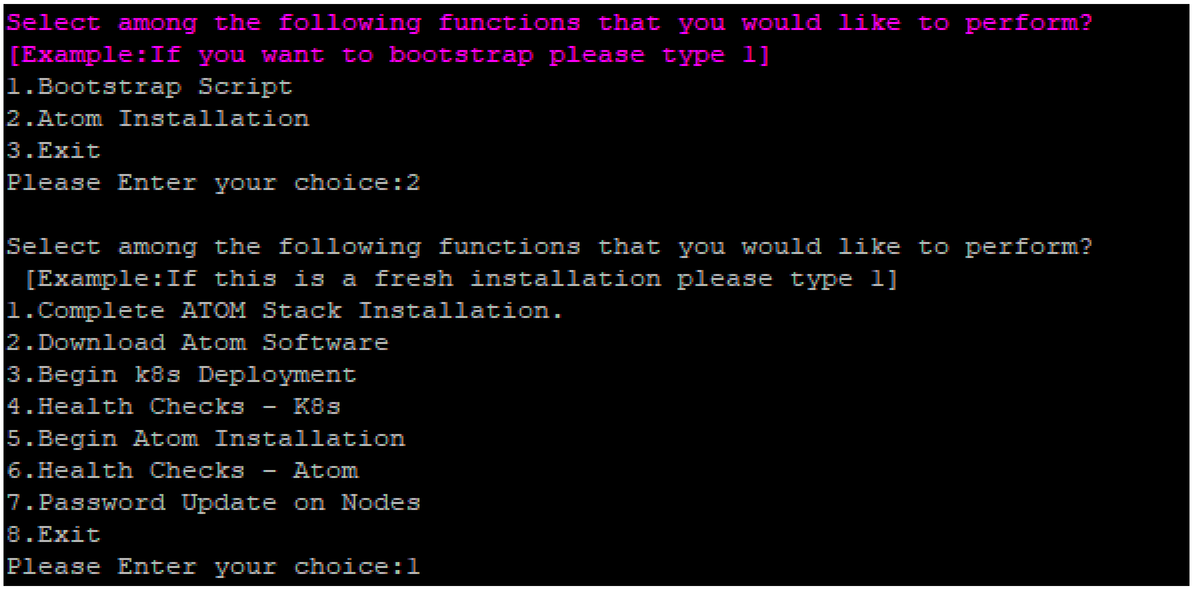


```
Current Progress : 2/3
Setting up Hostname on the node Initiated.

!!!!!!!!!!!!!!!!!!!!!!
Enter the host name : registry
Enter the ip address :172.16.26.105
Changes in hostname file is done.

These changes will be reflected after the system reboot.
Do you want to reboot the system ?(y/n)y
```

Hostname Configuration Details

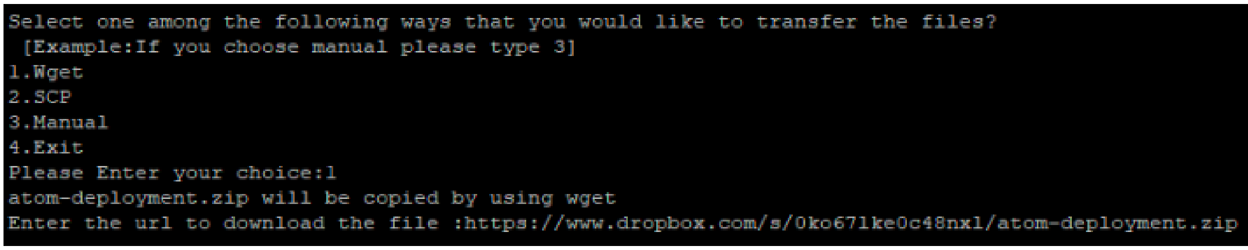
Once the bootstrap is complete proceed with the next steps. [Note: Hostname changes would be reflected on reboot only. Select yes to reboot if you wish to change the hostname]

8. Make sure Internet access is there from all the nodes.
9. After completion of the bootstrap process with VM reload, we are now ready to begin the atom installation process. Run the ***sudo python node_setup.py on Master Node*** for ATOM Installation.



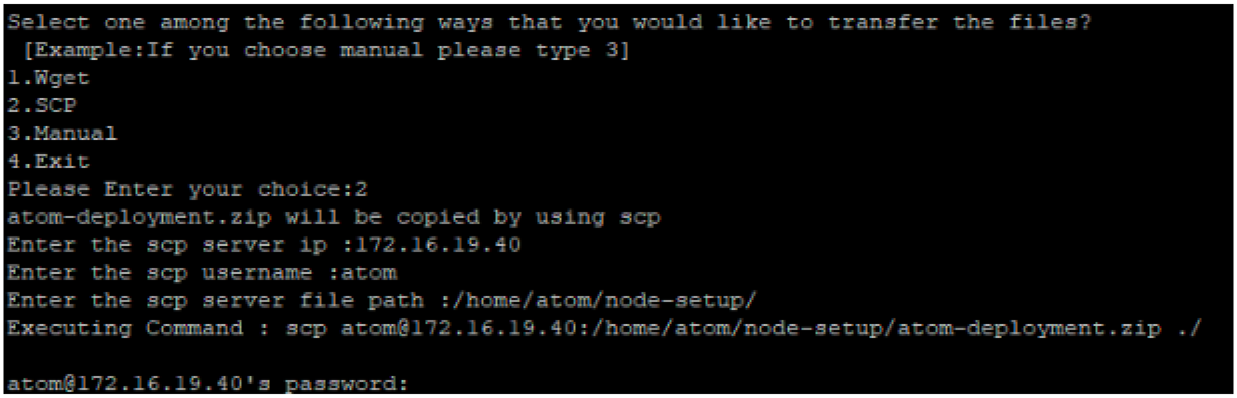
```
Select among the following functions that you would like to perform?
[Example:If you want to bootstrap please type 1]
1.Bootstrap Script
2.Atom Installation
3.Exit
Please Enter your choice:2

Select among the following functions that you would like to perform?
 [Example:If this is a fresh installation please type 1]
1.Complete ATOM Stack Installation.
2.Download Atom Software
3.Begin k8s Deployment
4.Health Checks - K8s
5.Begin Atom Installation
6.Health Checks - Atom
7.Password Update on Nodes
8.Exit
Please Enter your choice:1
```

10. Since it is a fresh install where the K8s cluster was also not created before, you can choose option1(Recommended) for Complete process of ZipDownload+K8s+ATOM Deployment (or) you can choose separately option 2(Only Download), option 3(only K8s) and option 5(ATOM).
    In case K8s cluster is already setup, one can directly proceed with just Atom software download and installation by selecting appropriate choices of 2, 5 respectively as in the ATOM Deployment section.
11. To download the **Atom Software Deployment zip** provided by Anuta Networks Support team we can use any of the methods as seen below:
    - Wget: Utility for non-interactive download of files from the Web. User needs to enter the link as input and the files would be downloaded and extracted automatically.



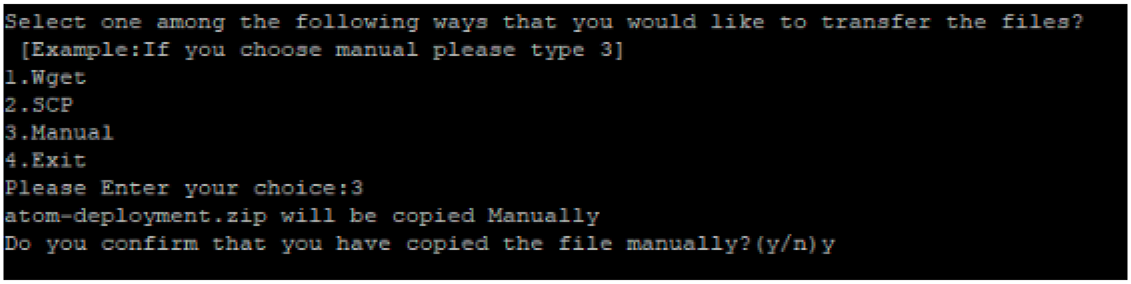
```
Select one among the following ways that you would like to transfer the files?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:1
atom-deployment.zip will be copied by using wget
Enter the url to download the file :https://www.dropbox.com/s/0ko67lke0c48nxl/atom-deployment.zip
```

- SCP: Securely transferring computer files between a local host and a remote host based on the Secure Shell (SSH) protocol.



```
Select one among the following ways that you would like to transfer the files?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:2
atom-deployment.zip will be copied by using scp
Enter the scp server ip :172.16.19.40
Enter the scp username :atom
Enter the scp server file path :/home/atom/node-setup/
Executing Command : scp atom@172.16.19.40:/home/atom/node-setup/atom-deployment.zip ./

atom@172.16.19.40's password:
```

- Manual: User can use any standard file transfer protocol to transfer the files on the home directory of the atom user.

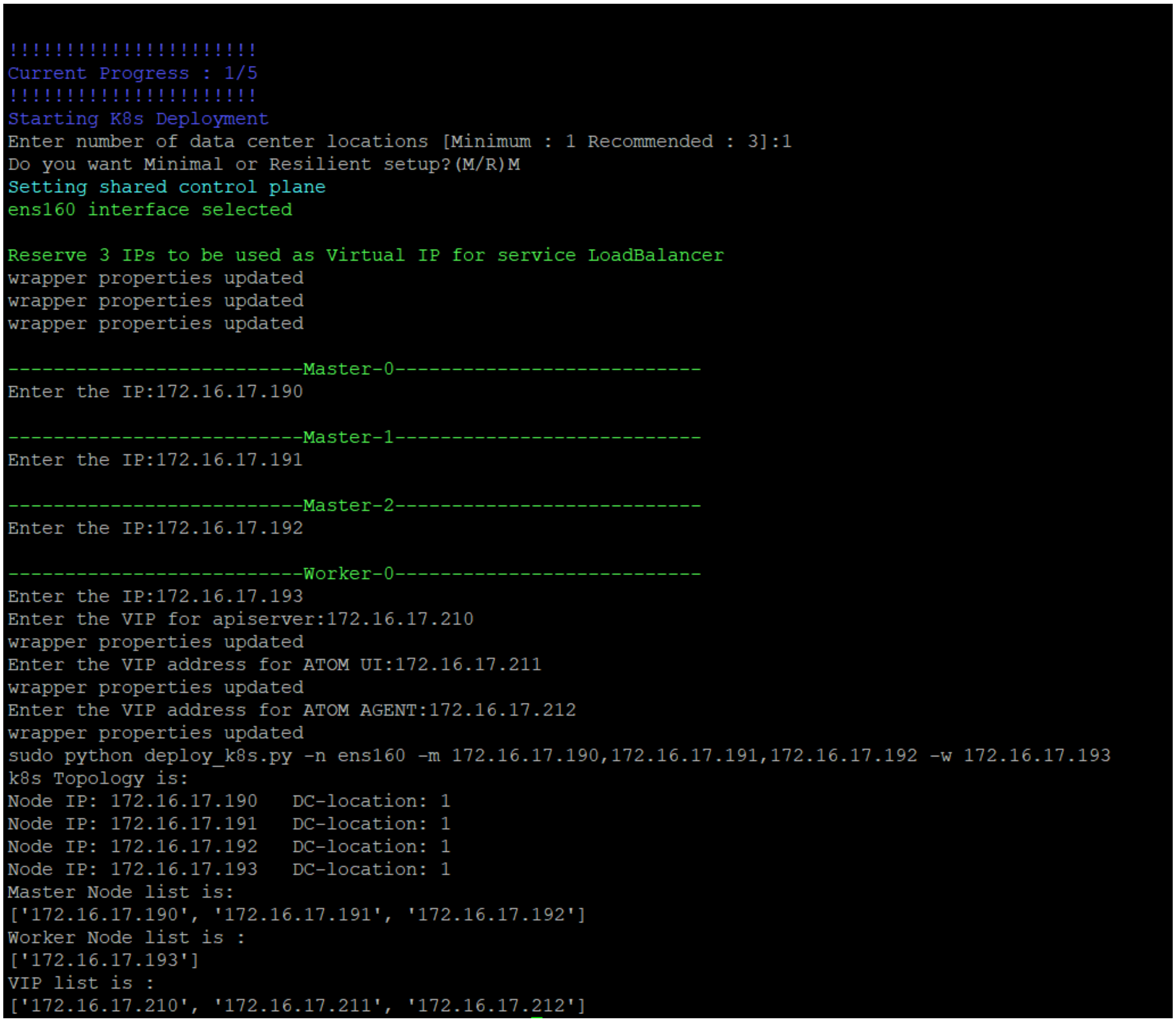


```
Select one among the following ways that you would like to transfer the files?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:3
atom-deployment.zip will be copied Manually
Do you confirm that you have copied the file manually?(y/n)y
```

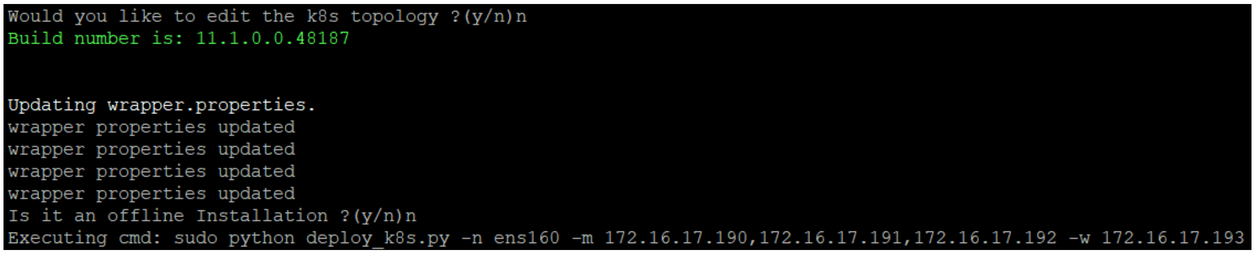
12. After the ATOM deployment zip installation files are copied on the Master Node we can begin with the K8s deployment. Depending on whether we want a minimal or resilient setup provide the inputs as shown below:

Minimal Setup deployment

Choose **1** as Data center locations, select **M** for minimal size**,** provide **VM IPs** and **Virtual IPs** info.



```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Enter number of data center locations [Minimum : 1 Recommended : 3]:1
Do you want Minimal or Resilient setup?(M/R)M
Setting shared control plane
ens160 interface selected

Reserve 3 IPs to be used as Virtual IP for service LoadBalancer
wrapper properties updated
wrapper properties updated
wrapper properties updated

--------------------------Master-0---------------------------
Enter the IP:172.16.17.190

--------------------------Master-1---------------------------
Enter the IP:172.16.17.191

--------------------------Master-2---------------------------
Enter the IP:172.16.17.192

--------------------------Worker-0---------------------------
Enter the IP:172.16.17.193
Enter the VIP for apiserver:172.16.17.210
wrapper properties updated
Enter the VIP address for ATOM UI:172.16.17.211
wrapper properties updated
Enter the VIP address for ATOM AGENT:172.16.17.212
wrapper properties updated
sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193
k8s Topology is:
Node IP: 172.16.17.190   DC-location: 1
Node IP: 172.16.17.191   DC-location: 1
Node IP: 172.16.17.192   DC-location: 1
Node IP: 172.16.17.193   DC-location: 1
Master Node list is:
['172.16.17.190', '172.16.17.191', '172.16.17.192']
Worker Node list is :
['172.16.17.193']
VIP list is :
['172.16.17.210', '172.16.17.211', '172.16.17.212']
```

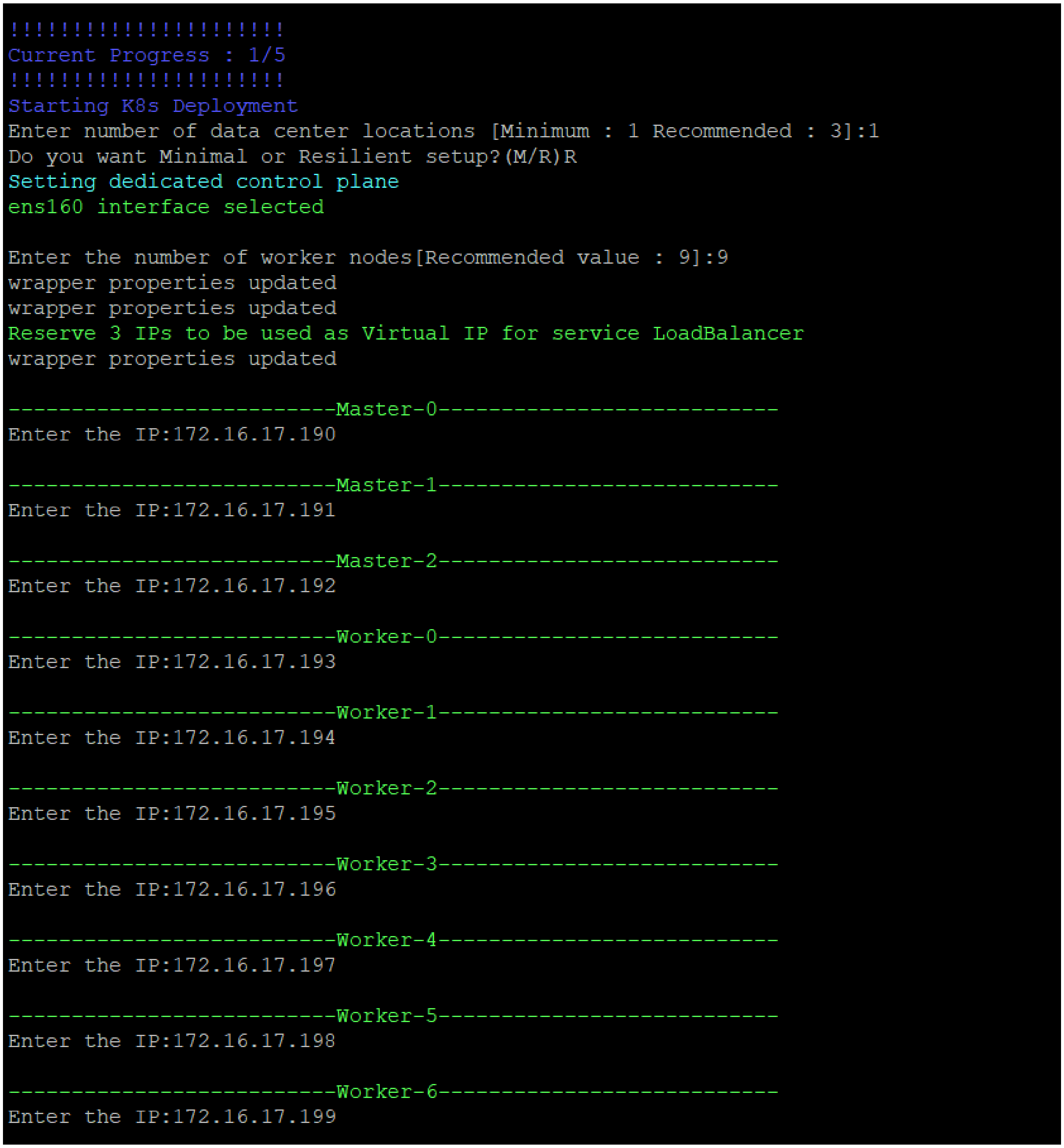


```
Would you like to edit the k8s topology ?(y/n)n
Build number is: 11.1.0.0.48187


Updating wrapper.properties.
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
Is it an offline Installation ?(y/n)n
Executing cmd: sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193
```

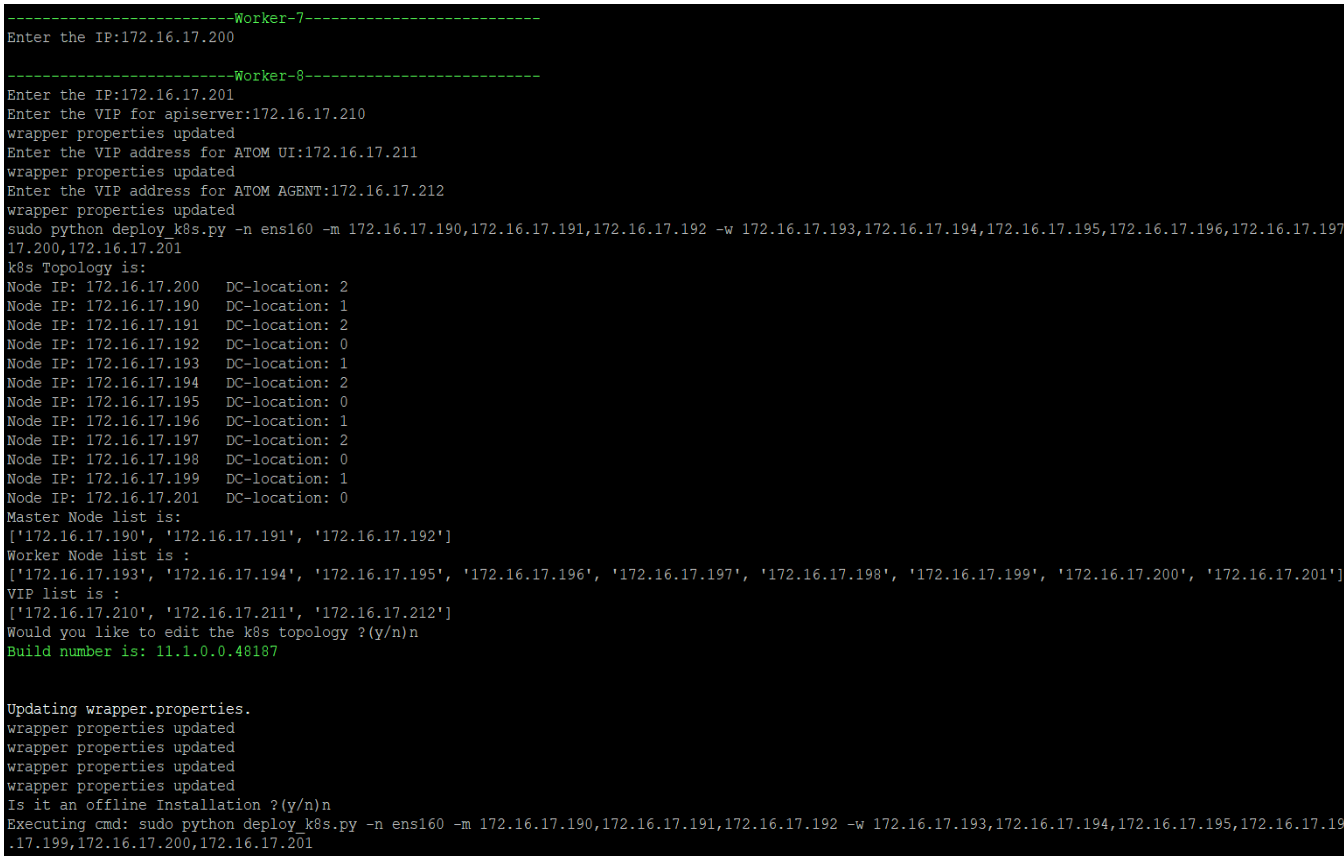
Resilient Setup deployment

1 Zone or DC location

Choose **1** as Data center locations, select **R** for resilient size**,** provide **VM IPs** and **Virtual IPs** info. Virtual DCs(3 nos) are created to maintain resiliency in this scenario.



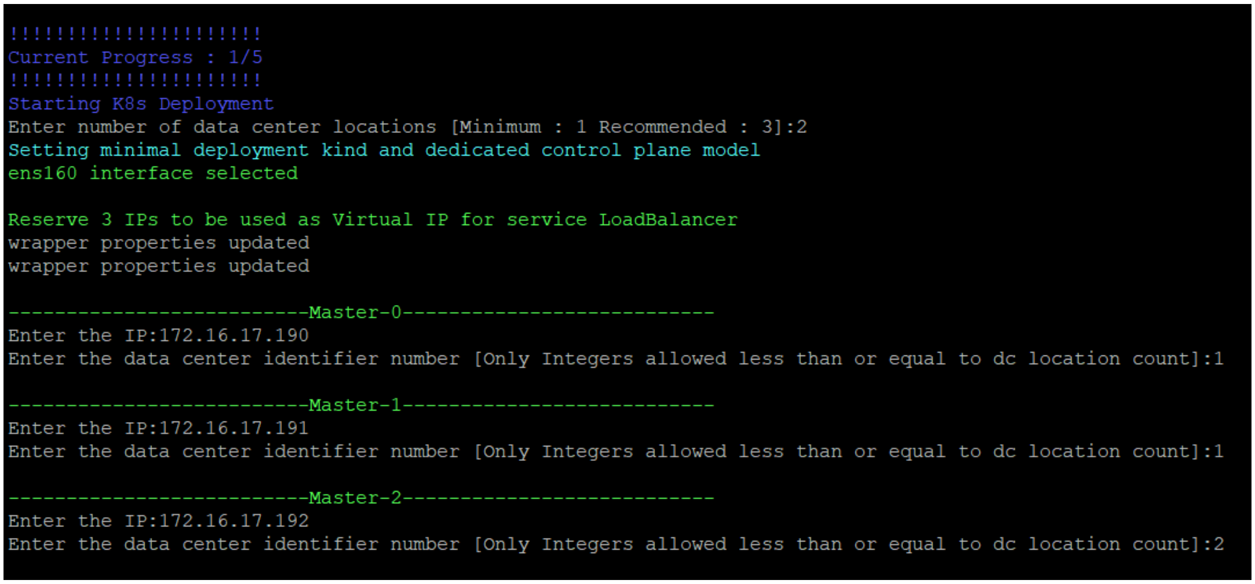
```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Enter number of data center locations [Minimum : 1 Recommended : 3]:1
Do you want Minimal or Resilient setup?(M/R)R
Setting dedicated control plane
ens160 interface selected

Enter the number of worker nodes[Recommended value : 9]:9
wrapper properties updated
wrapper properties updated
Reserve 3 IPs to be used as Virtual IP for service LoadBalancer
wrapper properties updated

--------------------------Master-0---------------------------
Enter the IP:172.16.17.190

--------------------------Master-1---------------------------
Enter the IP:172.16.17.191

--------------------------Master-2---------------------------
Enter the IP:172.16.17.192

--------------------------Worker-0---------------------------
Enter the IP:172.16.17.193

--------------------------Worker-1---------------------------
Enter the IP:172.16.17.194

--------------------------Worker-2---------------------------
Enter the IP:172.16.17.195

--------------------------Worker-3---------------------------
Enter the IP:172.16.17.196

--------------------------Worker-4---------------------------
Enter the IP:172.16.17.197

--------------------------Worker-5---------------------------
Enter the IP:172.16.17.198

--------------------------Worker-6---------------------------
Enter the IP:172.16.17.199
```



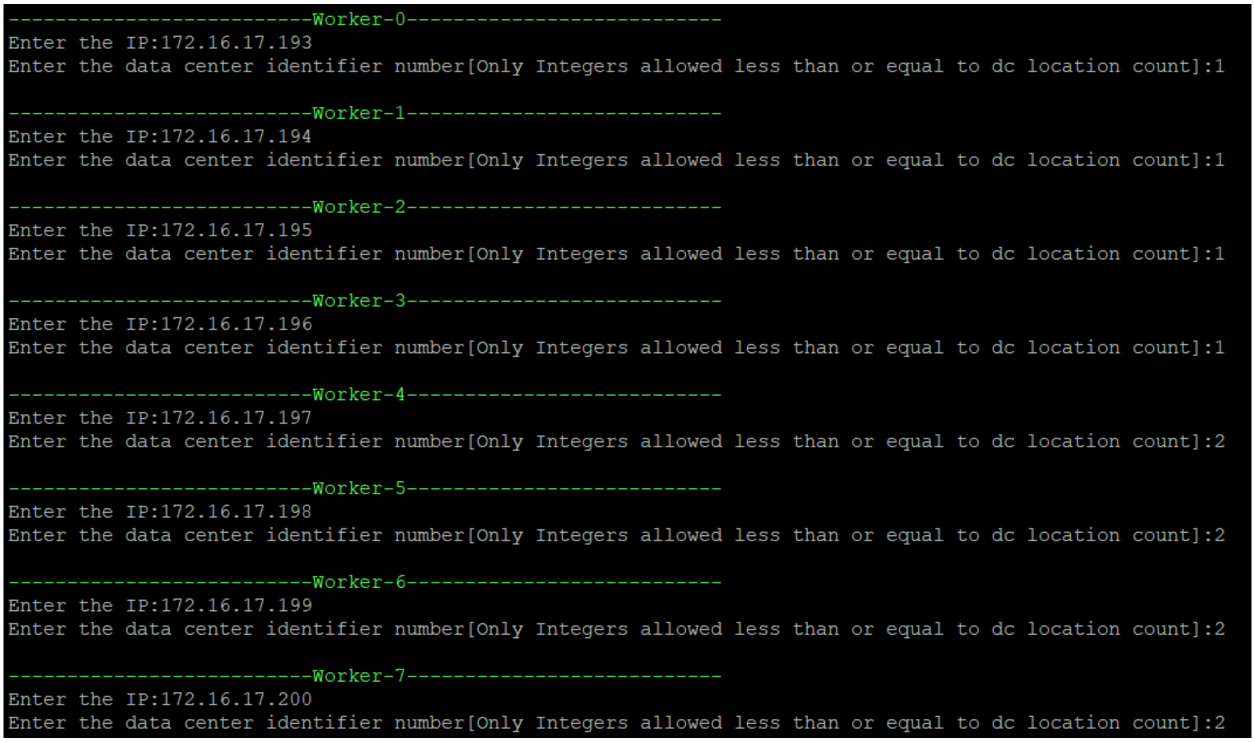
```
--------------------------Worker-7---------------------------
Enter the IP:172.16.17.200

--------------------------Worker-8---------------------------
Enter the IP:172.16.17.201
Enter the VIP for apiserver:172.16.17.210
wrapper properties updated
Enter the VIP address for ATOM UI:172.16.17.211
wrapper properties updated
Enter the VIP address for ATOM AGENT:172.16.17.212
wrapper properties updated
sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.195,172.16.17.196,172.16.17.197
17.200,172.16.17.201
k8s Topology is:
Node IP: 172.16.17.200   DC-location: 2
Node IP: 172.16.17.190   DC-location: 1
Node IP: 172.16.17.191   DC-location: 2
Node IP: 172.16.17.192   DC-location: 0
Node IP: 172.16.17.193   DC-location: 1
Node IP: 172.16.17.194   DC-location: 2
Node IP: 172.16.17.195   DC-location: 0
Node IP: 172.16.17.196   DC-location: 1
Node IP: 172.16.17.197   DC-location: 2
Node IP: 172.16.17.198   DC-location: 0
Node IP: 172.16.17.199   DC-location: 1
Node IP: 172.16.17.201   DC-location: 0
Master Node list is:
['172.16.17.190', '172.16.17.191', '172.16.17.192']
Worker Node list is :
['172.16.17.193', '172.16.17.194', '172.16.17.195', '172.16.17.196', '172.16.17.197', '172.16.17.198', '172.16.17.199', '172.16.17.200', '172.16.17.201']
VIP list is :
['172.16.17.210', '172.16.17.211', '172.16.17.212']
Would you like to edit the k8s topology ?(y/n)n
Build number is: 11.1.0.0.48187


Updating wrapper.properties.
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
Is it an offline Installation ?(y/n)n
Executing cmd: sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.195,172.16.17.19
.17.199,172.16.17.200,172.16.17.201
```

2 Zones or DC locations

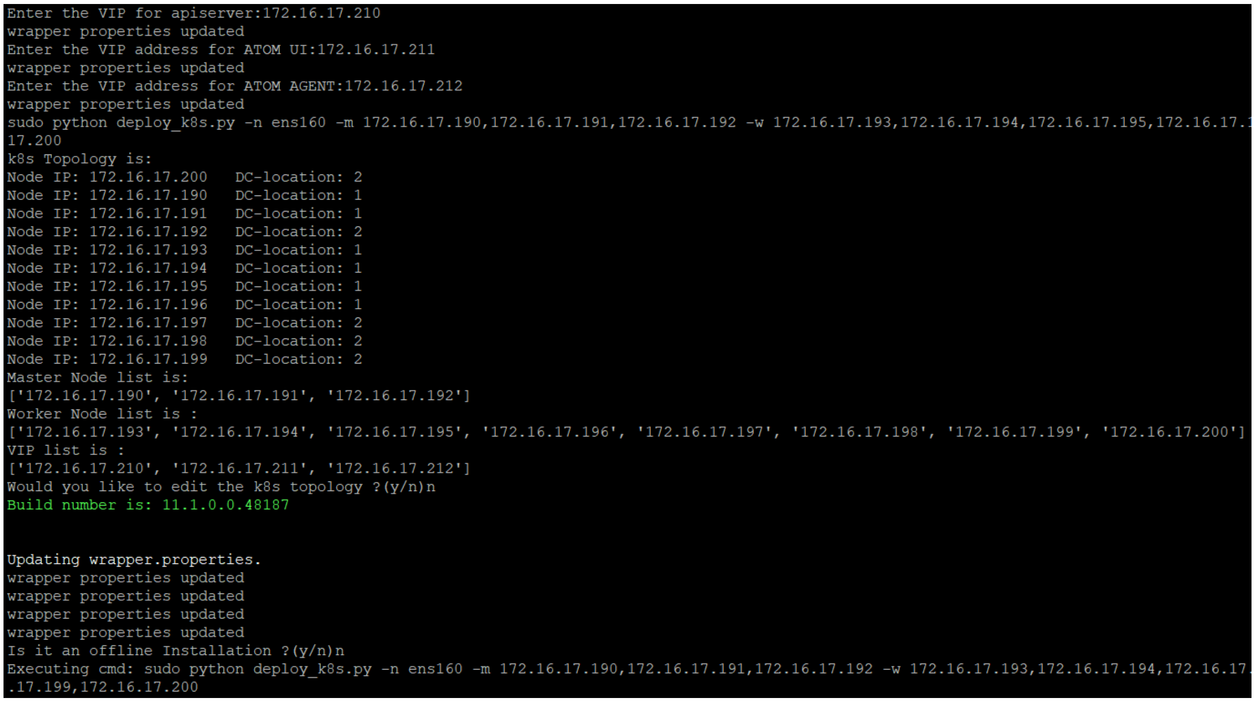
Choose **2** as Data center locations, select **R** for resilient size**,** provide **VM IPs** and **Virtual IPs** info. Input 2 master IPs for zone/DC 1 and 1 master IP for zone/DC 2. Provide 4 workers for each zone as shown below.



```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Enter number of data center locations [Minimum : 1 Recommended : 3]:2
Setting minimal deployment kind and dedicated control plane model
ens160 interface selected

Reserve 3 IPs to be used as Virtual IP for service LoadBalancer
wrapper properties updated
wrapper properties updated

--------------------------Master-0---------------------------
Enter the IP:172.16.17.190
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:1

--------------------------Master-1---------------------------
Enter the IP:172.16.17.191
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:1

--------------------------Master-2---------------------------
Enter the IP:172.16.17.192
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:2
```



```
--------------------------Worker-0---------------------------
Enter the IP:172.16.17.193
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-1---------------------------
Enter the IP:172.16.17.194
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-2---------------------------
Enter the IP:172.16.17.195
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-3---------------------------
Enter the IP:172.16.17.196
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-4---------------------------
Enter the IP:172.16.17.197
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-5---------------------------
Enter the IP:172.16.17.198
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-6---------------------------
Enter the IP:172.16.17.199
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-7---------------------------
Enter the IP:172.16.17.200
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2
```

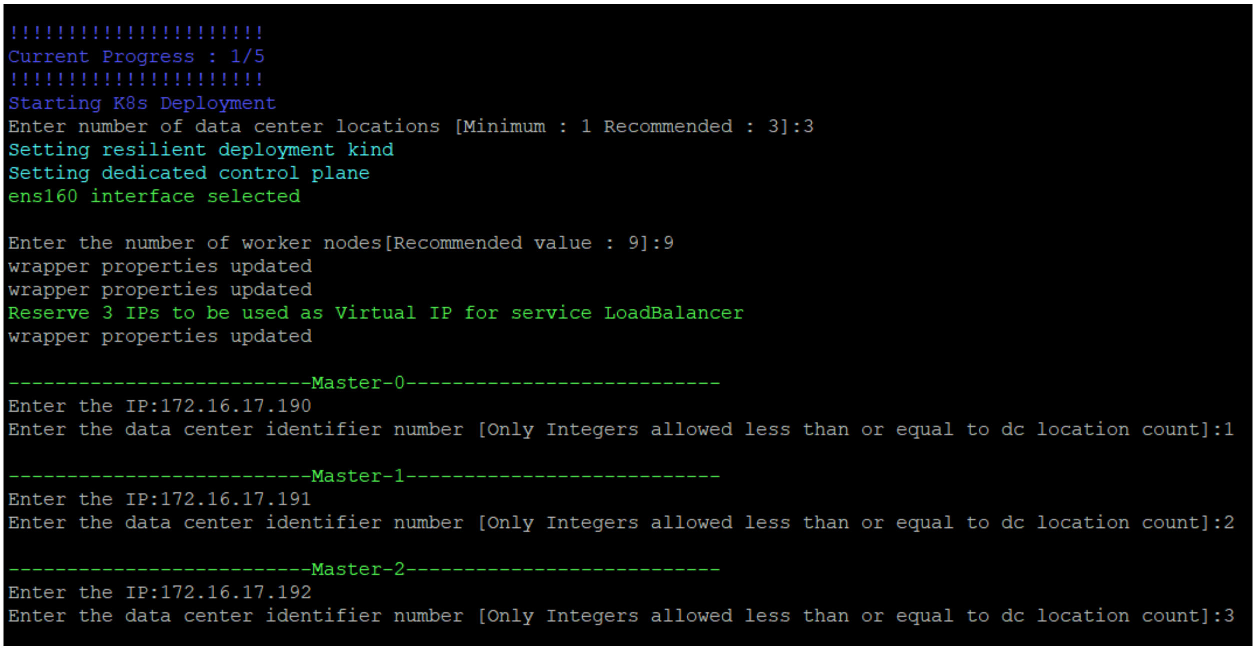


```
Enter the VIP for apiserver:172.16.17.210
wrapper properties updated
Enter the VIP address for ATOM UI:172.16.17.211
wrapper properties updated
Enter the VIP address for ATOM AGENT:172.16.17.212
wrapper properties updated
sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.195,172.16.17.1
17.200
k8s Topology is:
Node IP: 172.16.17.200   DC-location: 2
Node IP: 172.16.17.190   DC-location: 1
Node IP: 172.16.17.191   DC-location: 1
Node IP: 172.16.17.192   DC-location: 2
Node IP: 172.16.17.193   DC-location: 1
Node IP: 172.16.17.194   DC-location: 1
Node IP: 172.16.17.195   DC-location: 1
Node IP: 172.16.17.196   DC-location: 1
Node IP: 172.16.17.197   DC-location: 2
Node IP: 172.16.17.198   DC-location: 2
Node IP: 172.16.17.199   DC-location: 2
Master Node list is:
['172.16.17.190', '172.16.17.191', '172.16.17.192']
Worker Node list is :
['172.16.17.193', '172.16.17.194', '172.16.17.195', '172.16.17.196', '172.16.17.197', '172.16.17.198', '172.16.17.199', '172.16.17.200']
VIP list is :
['172.16.17.210', '172.16.17.211', '172.16.17.212']
Would you like to edit the k8s topology ?(y/n)n
Build number is: 11.1.0.0.48187


Updating wrapper.properties.
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
Is it an offline Installation ?(y/n)n
Executing cmd: sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.
.17.199,172.16.17.200
```

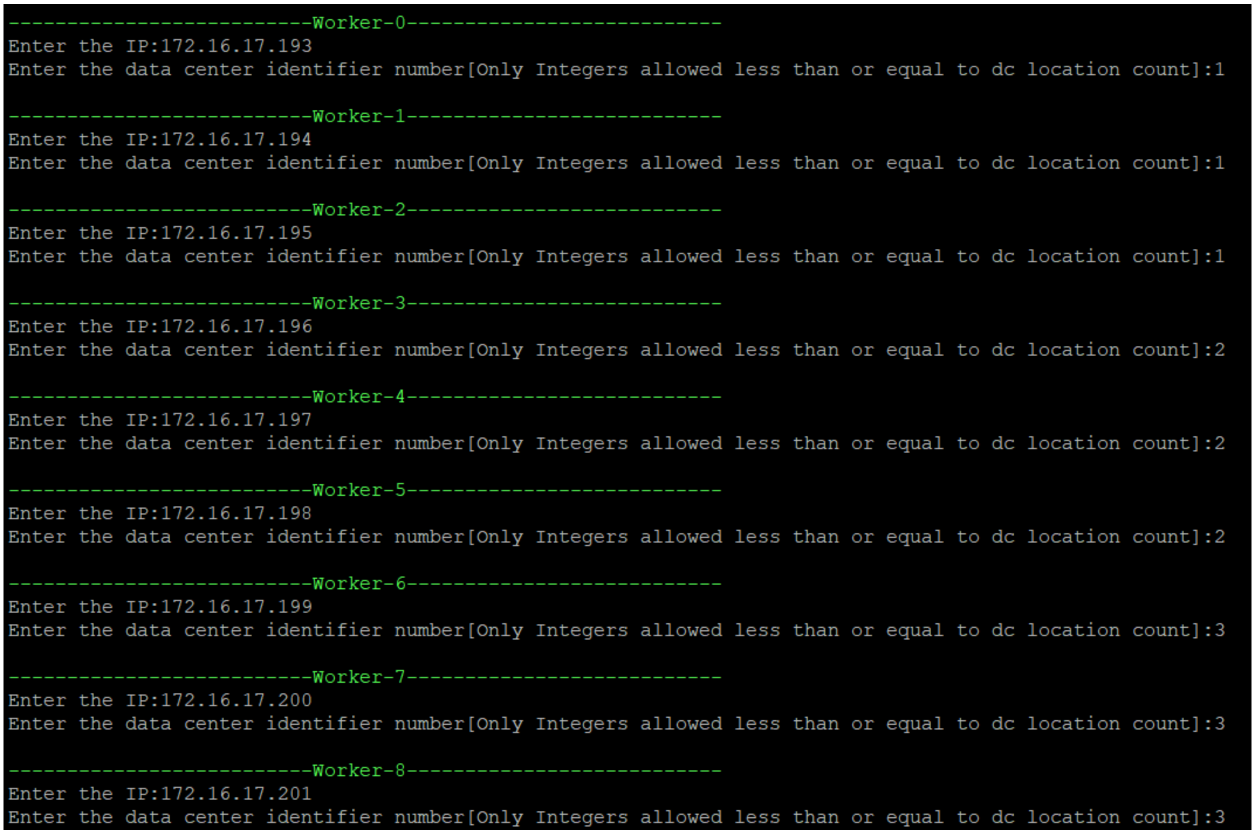
3 Zones or DC locations(Recommended)

Choose **3** as Data center locations, select **R** for resilient size**,** provide **VM IPs** and **Virtual IPs** info. Provide worker input as per zone/DC requirement.

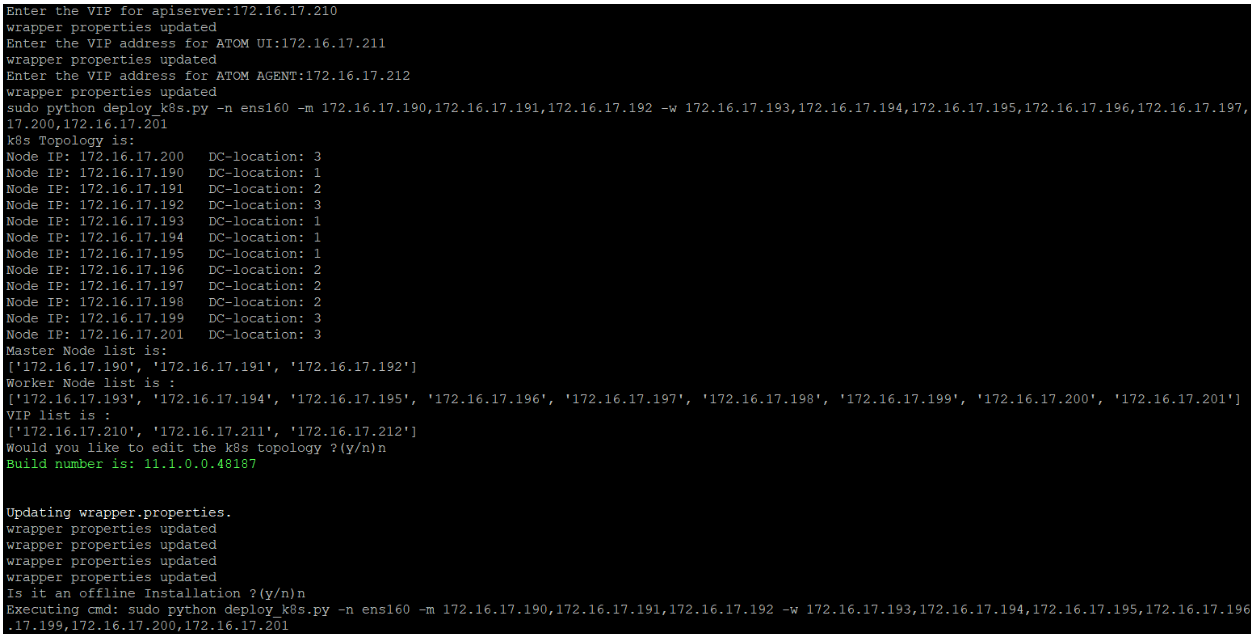
Note: Provide DC input alongside as shown below



```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Enter number of data center locations [Minimum : 1 Recommended : 3]:3
Setting resilient deployment kind
Setting dedicated control plane
ens160 interface selected

Enter the number of worker nodes[Recommended value : 9]:9
wrapper properties updated
wrapper properties updated
Reserve 3 IPs to be used as Virtual IP for service LoadBalancer
wrapper properties updated

--------------------------Master-0---------------------------
Enter the IP:172.16.17.190
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:1

--------------------------Master-1---------------------------
Enter the IP:172.16.17.191
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:2

--------------------------Master-2---------------------------
Enter the IP:172.16.17.192
Enter the data center identifier number [Only Integers allowed less than or equal to dc location count]:3
```



```
--------------------------Worker-0---------------------------
Enter the IP:172.16.17.193
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-1---------------------------
Enter the IP:172.16.17.194
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-2---------------------------
Enter the IP:172.16.17.195
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:1

--------------------------Worker-3---------------------------
Enter the IP:172.16.17.196
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-4---------------------------
Enter the IP:172.16.17.197
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-5---------------------------
Enter the IP:172.16.17.198
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:2

--------------------------Worker-6---------------------------
Enter the IP:172.16.17.199
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:3

--------------------------Worker-7---------------------------
Enter the IP:172.16.17.200
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:3

--------------------------Worker-8---------------------------
Enter the IP:172.16.17.201
Enter the data center identifier number[Only Integers allowed less than or equal to dc location count]:3
```



```
Enter the VIP for apiserver:172.16.17.210
wrapper properties updated
Enter the VIP address for ATOM UI:172.16.17.211
wrapper properties updated
Enter the VIP address for ATOM AGENT:172.16.17.212
wrapper properties updated
sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.195,172.16.17.196,172.16.17.197,
17.200,172.16.17.201
k8s Topology is:
Node IP: 172.16.17.200   DC-location: 3
Node IP: 172.16.17.190   DC-location: 1
Node IP: 172.16.17.191   DC-location: 2
Node IP: 172.16.17.192   DC-location: 3
Node IP: 172.16.17.193   DC-location: 1
Node IP: 172.16.17.194   DC-location: 1
Node IP: 172.16.17.195   DC-location: 1
Node IP: 172.16.17.196   DC-location: 2
Node IP: 172.16.17.197   DC-location: 2
Node IP: 172.16.17.198   DC-location: 2
Node IP: 172.16.17.199   DC-location: 3
Node IP: 172.16.17.201   DC-location: 3
Master Node list is:
['172.16.17.190', '172.16.17.191', '172.16.17.192']
Worker Node list is :
['172.16.17.193', '172.16.17.194', '172.16.17.195', '172.16.17.196', '172.16.17.197', '172.16.17.198', '172.16.17.199', '172.16.17.200', '172.16.17.201']
VIP list is :
['172.16.17.210', '172.16.17.211', '172.16.17.212']
Would you like to edit the k8s topology ?(y/n)n
Build number is: 11.1.0.0.48187


Updating wrapper.properties.
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
Is it an offline Installation ?(y/n)n
Executing cmd: sudo python deploy_k8s.py -n ens160 -m 172.16.17.190,172.16.17.191,172.16.17.192 -w 172.16.17.193,172.16.17.194,172.16.17.195,172.16.17.196
.17.199,172.16.17.200,172.16.17.201
```

13. Above will create K8s cluster among Master and Worker Nodes spread across Esxi/Locations which have reachability.
14. On a different shell terminal to master node, you can as well verify the nodes cluster formation using the command **"kubectl get nodes**" and verify labels using the command "**kubectl get nodes --show-labels**"
15. If the deployment model is selected as offline then provide the Registry IP address and Project repo name as provided during the docker registry installation.
16. As now the Kubernetes cluster's creation is done and it is ready for ATOM deployment. If option1 is chosen in step8 then, "ATOM Deployment" will happen next in the process or it can be invoked separately as well with option5.

ATOM Deployment

After ensuring that the prerequisites are taken care as described in the section, "Prerequisites for Deploying ATOM", perform the following steps:

1. For Minimal or Resilient HA setup, ensure that K8s cluster is formed and the worker nodes are labelled properly as below using the command "**kubectl get nodes --show-labels**".

   For Minimal setup and Resilient HA single DC setup.
   elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb

   For2 DC/Zones:
   Zone 1:

elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb,topology.kubernetes.io/zone=dc-1
Zone 2:
elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb,topology.kubernetes.io/zone=dc-2

For Resilient-HA setup in 3 DC/Zones:
Zone 1:
elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb,topology.kubernetes.io/zone=dc-1
Zone 2:
elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb,topology.kubernetes.io/zone=dc-2
Zone 3:
elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb,topology.kubernetes.io/zone=dc-3

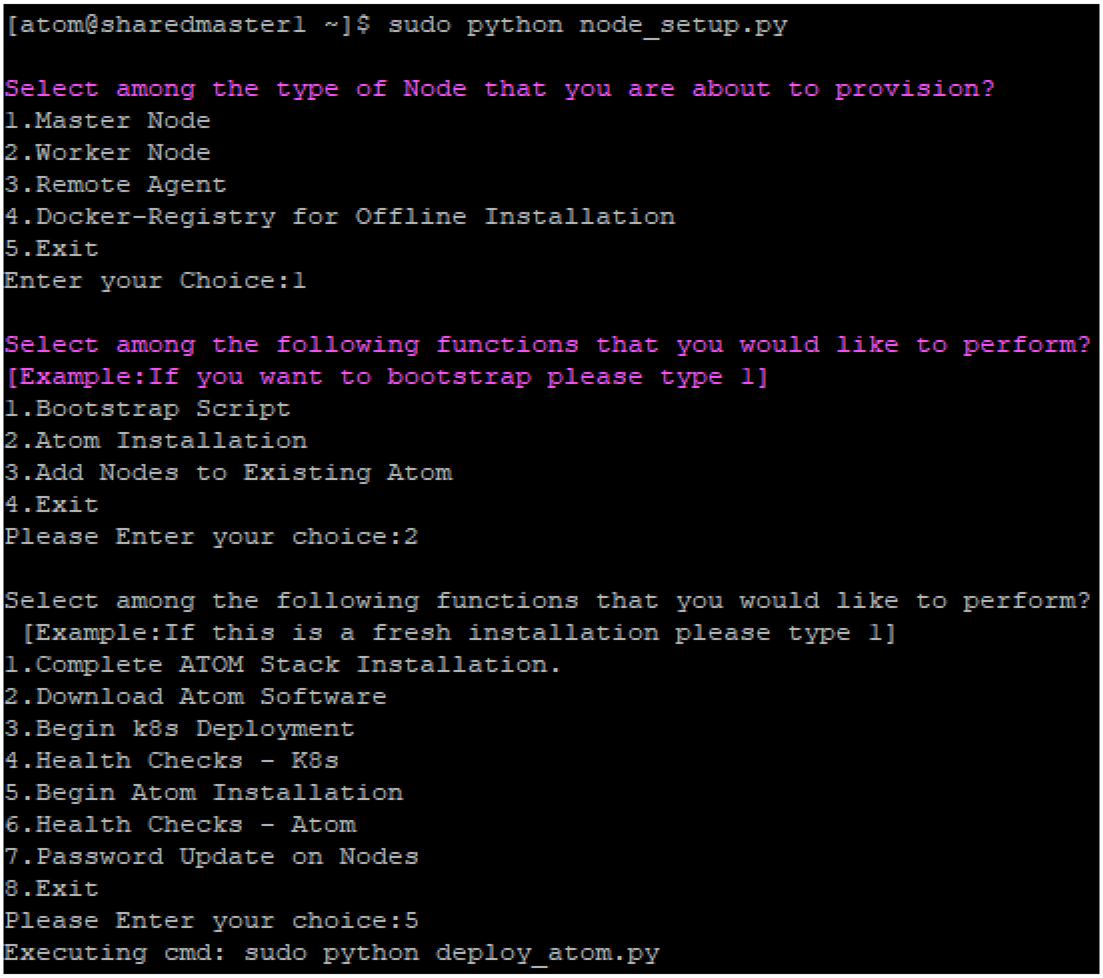
To label a node use below command:

```
kubectl label node <node-name> <label_name>=deploy
```

Note: Make sure you see label dc1, dc2 and dc3 appropriately based on the datacenter where it is present for Resilient HA setup. For scale Worker Nodes also the labelling approach remains the same as above.

2. To download the **Atom Software Deployment zip**(in case not done before) provided by Anuta Networks Support team, you can use any of the download methods described in the section [New Kubernetes cluster]() step 9-11.

3. On the master node of the Kubernetes cluster, if option1 was chosen at step9 of [New Kubernetes cluster]() or if option5 chosen to trigger ATOM deployment separately, then all of the ATOM application components/microservices will get deployed.



```
[atom@sharedmaster1 ~]$ sudo python node_setup.py

Select among the type of Node that you are about to provision?
1.Master Node
2.Worker Node
3.Remote Agent
4.Docker-Registry for Offline Installation
5.Exit
Enter your Choice:1

Select among the following functions that you would like to perform?
[Example:If you want to bootstrap please type 1]
1.Bootstrap Script
2.Atom Installation
3.Add Nodes to Existing Atom
4.Exit
Please Enter your choice:2

Select among the following functions that you would like to perform?
 [Example:If this is a fresh installation please type 1]
1.Complete ATOM Stack Installation.
2.Download Atom Software
3.Begin k8s Deployment
4.Health Checks - K8s
5.Begin Atom Installation
6.Health Checks - Atom
7.Password Update on Nodes
8.Exit
Please Enter your choice:5
Executing cmd: sudo python deploy_atom.py
```

NOTE: The order in which the ATOM components should be deployed is already defined in the scripts.

OPTIONAL: If a different namespace (instead of atom namespace) needs to be used, then do changes in functional_minimal.yaml file:

```
usernamespace:
 enabled: false
 namespace: <mynamespace>

namespace: <mynamespace>
```

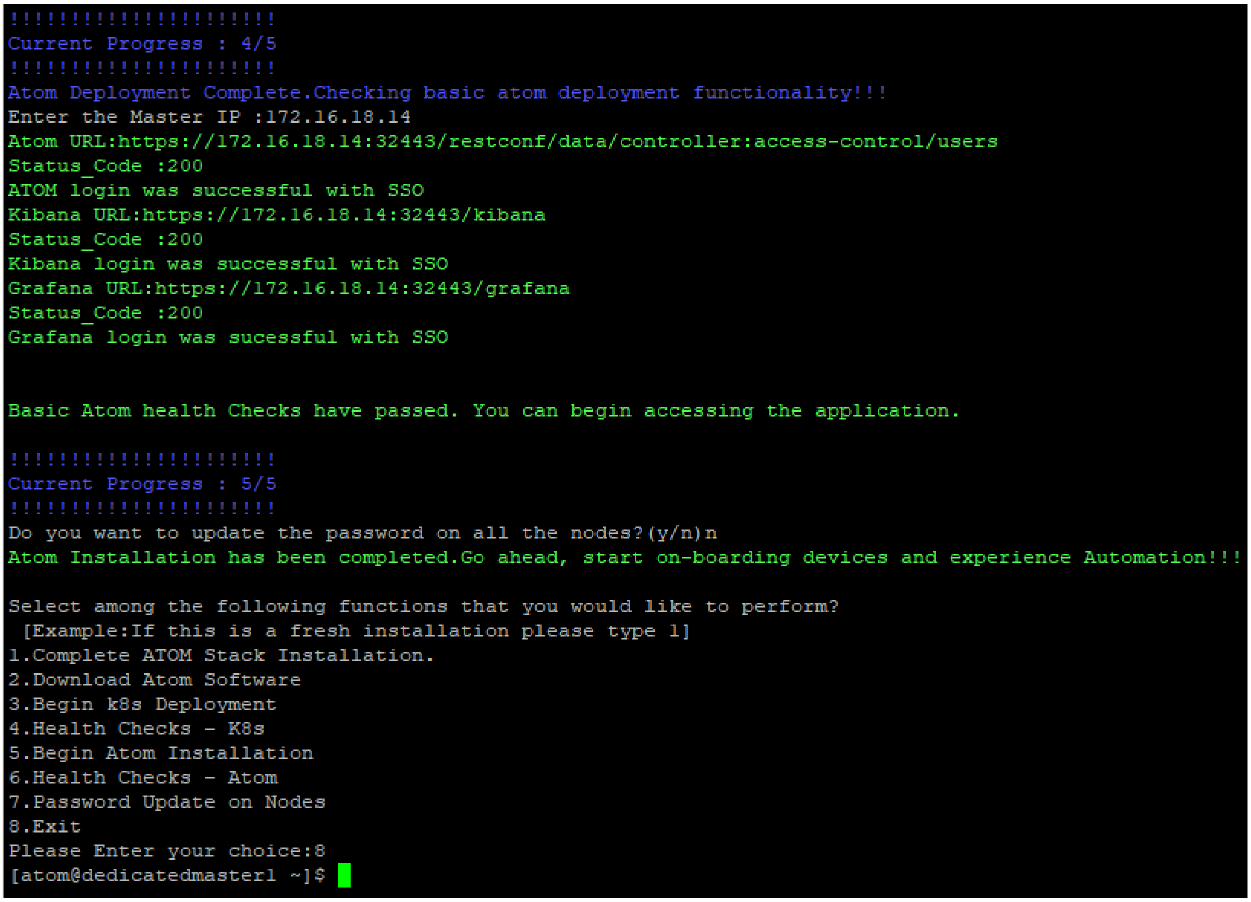
A successful ATOM deployment of the components using Helm will have sample output like below:

```
node resources met, proceeding..
master ip fetched from wrapper.properties172.16.18.5
quay
anuta docker registry secret was not found, creating it
helm check is successful
```

```
Folders creating done.
PV creating done.
All module check is successful
deploying Linstor charts using piraeus-operator
DB pods deployed
Helm chart haproxy got successfully deployed
Helm chart keycloak skipped
Helm chart infra-kibana got successfully deployed
Helm chart haproxy-gw got successfully deployed
Helm chart dashboard got successfully deployed
Helm chart oauth2 skipped
Helm chart lb got successfully deployed
Helm chart infra-grafana got successfully deployed
Helm chart infra-distributed-db-webconsole got successfully deployed
Helm chart infra-logstash got successfully deployed
Helm chart broker got successfully deployed
Helm chart zookeeper got successfully deployed
Helm chart infra-distributed-db-webagent got successfully deployed
Helm chart infra-log-forwarder got successfully deployed
Helm chart elasticsearch-config got successfully deployed
Helm chart schema-repo got successfully deployed
Helm chart infra-elasticsearch got successfully deployed
Helm chart infra-distributed-db got successfully deployed
returncode is  0
DB pods deployed
Helm chart infra-tsdb-monitoring got successfully deployed
Helm chart minio got successfully deployed
Helm chart thanos got successfully deployed
Helm chart atom-workflow-engine got successfully deployed
Helm chart atom-inventory-mgr got successfully deployed
Helm chart atom-isim got successfully deployed
Helm chart kafka-operator got successfully deployed
Helm chart atom-pnp-server got successfully deployed
Helm chart atom-core got successfully deployed
Helm chart atom-qs got successfully deployed
Helm chart atom-agent-proxy got successfully deployed
Helm chart atom-scheduler got successfully deployed
Helm chart atom-sysmgr got successfully deployed
Helm chart atom-agent got successfully deployed
Helm chart atom-telemetry-engine got successfully deployed
Helm chart atom-ml got successfully deployed
Helm chart atom-frontend got successfully deployed
Helm chart infra-glowroot got successfully deployed
Helm chart burrow got successfully deployed
Helm chart jaeger-tracing got successfully deployed
Helm chart kafka-control got successfully deployed
Helm chart kafka-manager got successfully deployed
Helm chart infra-web-proxy got successfully deployed
Helm chart infra-tsdb got successfully deployed
Helm chart modsecurity got successfully deployed
Supplied atom as namespace
SSO URLS for application endpoints are:
ATOM UI ==> https://172.16.18.20
KIBANA UI ==> https://172.16.18.20/kibana/
GRAFANA UI ==> https://172.16.18.20/grafana/
GLOWROOT UI ==> https://172.16.18.20/glowroot/
```

```
K8S UI ==> https://172.16.18.20/k8s/
KAFKA MANAGER UI ==> https://172.16.18.20/kafka-manager/
KEYCLOAK_URL ==> https://172.16.18.20/auth
TSDB_URL ==> https://172.16.18.20/prometheus
THANOS_URL ==> https://172.16.18.20/thanos
TSDB_MONITORING_URL ==> https://172.16.18.20/prometheus-atom
REMOTE AGENT URL ==> :7000
ZTP URL ==>

('atom_fqdn = ', '')
sh /home/atom/atom-deployment/scripts/get_urls.sh atom
172.16.18.18
https://172.16.18.18:
Keycloak is active
Fetching token from admin-cli
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJPUmg4MWFSNUpOS21nelB6
aFNmSEVlYnQ5ekZRVVYyaVFwWG5hYmhLNERRIn0.eyJleHAiOjE2NDU1MDQ5NjYsImlhdCI6
MTY0NTUwNDkwNiwianRpIjoiZjQ2ZTBlZGYtM2FkMi00ODJmLTljYzktY2I2YzcyNzA4YTM1
IiwiaXNzIjoiaHR0cHM6Ly8xNzIuMTYuMTguMTgvYXV0aC9yZWFsbXMvbWFzdGVyIiwic3Vi
IjoiNmUzYWEyZmItOGI1NS00ZjlmLTliMWQtZjFmZTBmZTdkZjBiIiwidHlwIjoiQmVhcmVy
IiwiYXpwIjoiYWRtaW4tY2xpIiwic2Vzc2lvbl9zdGF0ZSI6IjkxMzliOTQzLWI5NjAtNGJl
NS05MWYzLTI3ZGE1NmIzMzgyNiIsImFjciI6IjEiLCJzY29wZSI6ImVtYWlsIHByb2ZpbGUi
LCJlbWFpbF92ZXJpZmllZCI6ZmFsc2UsInByZWZlcnJlZF91c2VybmFtZSI6ImFkbWluIn0.
Shom2O7DkYkS9aI0MsdUitY7mSlDHUtSgsMiZiWwPNHvionKLFNVeE4ynhP8sl3k3KLZQ5UJ
MbhORvKNorxLQqCLIZNZONhtFnxEY9OQLXQKkE29xORCPkpj1ooDISEU2Wj5quLkEpSh8BsP
pN9bCcNeJKqabwbIBCdo8wGNFa8WrL5M34jNIMmKR-h2e6UrZMX9LpOpKY8B5z6w7kRQ3LwK
f700etth24WMw4qlYkdYlk57OFoPcWa8PvcSA0_j52iva1Bv4vVE4EPfeR46bbhSillngBTS
WA5ycuhyZPcwHJOpNE3GzkgCKeyygz9us7_BwYFLQ2cwS2Q13Qn-lQ
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoints reachability check in progress...
Endpoint                       Status
THANOS_URL                     REACHABLE
GLOWROOT UI                    REACHABLE
KEYCLOAK_URL                   REACHABLE
TSDB_MONITORING_URL            REACHABLE
K8S UI                         REACHABLE
GRAFANA UI                     REACHABLE
KIBANA UI                      REACHABLE
ATOM UI                        REACHABLE
KAFKA MANAGER UI               NOT REACHABLE
TSDB_URL                       REACHABLE
```

4. After completing atom-deployment.Again run node_setup.py script and select 6th option and give master ip for checking basic functionality ,next select 7th option if you want to change passwords.



```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 4/5
!!!!!!!!!!!!!!!!!!!!!!
Atom Deployment Complete.Checking basic atom deployment functionality!!!
Enter the Master IP :172.16.18.14
Atom URL:https://172.16.18.14:32443/restconf/data/controller:access-control/users
Status_Code :200
ATOM login was successful with SSO
Kibana URL:https://172.16.18.14:32443/kibana
Status_Code :200
Kibana login was successful with SSO
Grafana URL:https://172.16.18.14:32443/grafana
Status_Code :200
Grafana login was sucessful with SSO


Basic Atom health Checks have passed. You can begin accessing the application.

!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 5/5
!!!!!!!!!!!!!!!!!!!!!!
Do you want to update the password on all the nodes?(y/n)n
Atom Installation has been completed.Go ahead, start on-boarding devices and experience Automation!!!

Select among the following functions that you would like to perform?
 [Example:If this is a fresh installation please type 1]
1.Complete ATOM Stack Installation.
2.Download Atom Software
3.Begin k8s Deployment
4.Health Checks - K8s
5.Begin Atom Installation
6.Health Checks - Atom
7.Password Update on Nodes
8.Exit
Please Enter your choice:8
[atom@dedicatedmaster1 ~]$
```

5. A summary of access URLs for various components deployed will be displayed after installation and if required can be obtained anytime by executing following command in scripts folder

```
# cd scripts
# sh get_urls.sh
```

The output will be similar to below

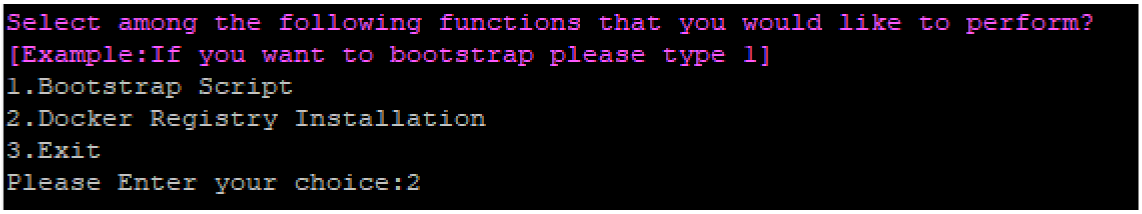
```
Supplied atom as namespace
SSO URLS for application endpoints are:
ATOM UI ==> https://172.16.18.18
KIBANA UI ==> https://172.16.18.18/kibana/
GRAFANA UI ==> https://172.16.18.18/grafana/
GLOWROOT UI ==> https://172.16.18.18/glowroot/
K8S UI ==> https://172.16.18.18/k8s/
KAFKA MANAGER UI ==> https://172.16.18.18/kafka-manager/
KEYCLOAK_URL ==> https://172.16.18.18/auth
TSDB_URL ==> https://172.16.18.18/prometheus
THANOS_URL ==> https://172.16.18.18/thanos
TSDB_MONITORING_URL ==> https://172.16.18.18/prometheus-atom
```

Docker registry for Offline deployment

ATOM can be deployed offline using the locally hosted docker registry. Docker images have to be pulled from a locally available registry to the respective nodes for atom deployment.

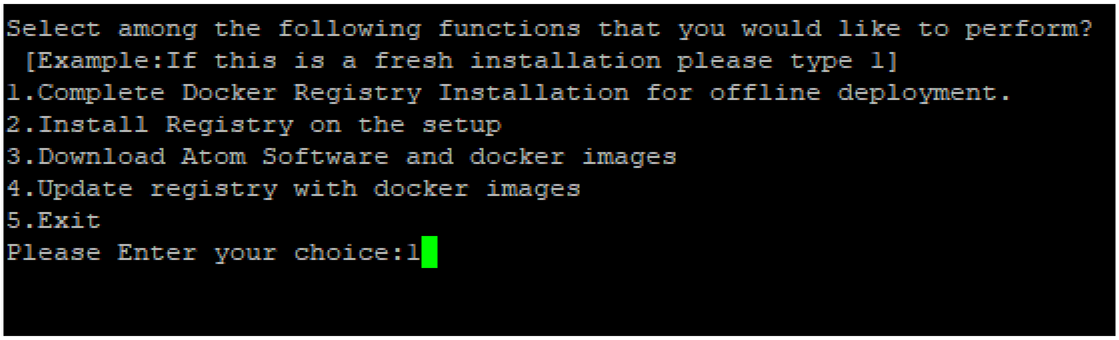
Verify that you have imported the shared Anuta docker-registry OVA template into your VMware vCenter.

1. The specs for docker registry VM will be 4CPU/32GB RAM/300GB SSD/1 NIC
2. Log into the VM using default creds atom/secret@123.
3. For bootstrapping the node with basic Interface, DNS and NTP configs run the node_setup.py which is present in the home directory using sudo privileges as described in the section New Kubernetes cluster
4. After completion of the bootstrap process we are now ready to begin the Docker registry installation process. Run node_setup.py script and select Docker registry installation by entering 2 when prompted for choice.



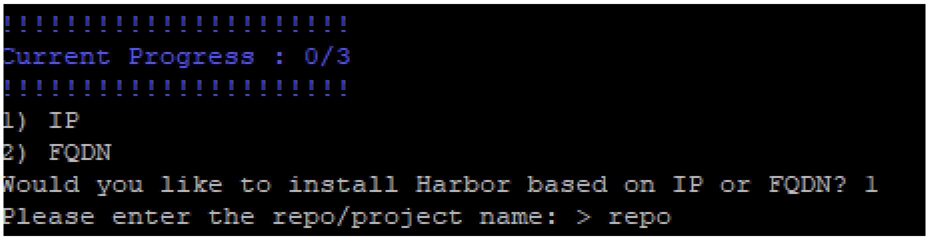
```
Select among the following functions that you would like to perform?
[Example:If you want to bootstrap please type 1]
1.Bootstrap Script
2.Docker Registry Installation
3.Exit
Please Enter your choice:2
```

5. For a fresh install we can select “Complete Docker Registry Installation for offline Deployment” option by entering 1. If required we can perform each of the other steps in the exact order individually.In case of failure, the user can retry by giving appropriate options where the process had failed.



```
Select among the following functions that you would like to perform?
 [Example:If this is a fresh installation please type 1]
1.Complete Docker Registry Installation for offline deployment.
2.Install Registry on the setup
3.Download Atom Software and docker images
4.Update registry with docker images
5.Exit
Please Enter your choice:1
```

6. Provide the IP option using “1” or use hostname if they can be resolved. Give the project name which would serve the purpose of repo name.It needs to be provided at a later stage so do make note of it.



```
!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 0/3
!!!!!!!!!!!!!!!!!!!!!!
1) IP
2) FQDN
Would you like to install Harbor based on IP or FQDN? 1
Please enter the repo/project name: > repo
```

Default login for registry will be **admin/admin (http:<registry-ip>)**

Output of the above process may take time and would look as follows:

```
Redirecting to /bin/systemctl restart docker.service
harbor/harbor.v2.2.1.tar.gz
harbor/prepare
harbor/LICENSE
harbor/install.sh
harbor/common.sh
harbor/harbor.yml.tmpl
prepare base dir is set to /home/atom/harbor
Unable to find image 'goharbor/prepare:v2.2.1' locally
docker: Error response from daemon: Get https://registry-1.docker.io/v2/: dial tcp:
lookup registry-1.docker.io on 8.8.8.8:53: read udp
172.16.26.105:53734->8.8.8.8:53: i/o timeout.
See 'docker run --help'.

[Step 0]: checking if docker is installed ...

Note: docker version: 20.10.5

[Step 1]: checking docker-compose is installed ...

Note: docker-compose version: 1.25.5

[Step 2]: loading Harbor images ...
23e1126e5547: Loading layer [==================================================>]
34.51MB/34.51MB
0a791fa5d10a: Loading layer [==================================================>]
6.241MB/6.241MB
478208477097: Loading layer [==================================================>]
4.096kB/4.096kB
a31ccda4a655: Loading layer [==================================================>]
3.072kB/3.072kB
70f59ceb330c: Loading layer [==================================================>]
28.3MB/28.3MB
ef395db1a0f0: Loading layer [==================================================>]
11.38MB/11.38MB
fb2e075190ca: Loading layer [==================================================>]
40.5MB/40.5MB
Loaded image: goharbor/trivy-adapter-photon:v2.2.1
c3a4c23b7b9c: Loading layer [==================================================>]
8.075MB/8.075MB
00f54a3b0f73: Loading layer [==================================================>]
3.584kB/3.584kB
afc25040e33f: Loading layer [==================================================>]
2.56kB/2.56kB
```

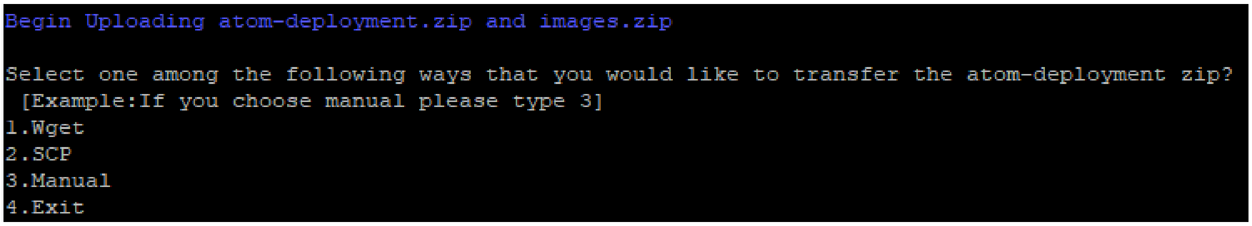
```
edb7c59d9116: Loading layer [==================================================>]
61.03MB/61.03MB
e5405375a1be: Loading layer [==================================================>]
61.85MB/61.85MB
Loaded image: goharbor/harbor-jobservice:v2.2.1
ab7d4d8af822: Loading layer [==================================================>]
4.937MB/4.937MB
8eb4015cb760: Loading layer [==================================================>]
4.096kB/4.096kB
4be492c354d6: Loading layer [==================================================>]
3.072kB/3.072kB
ea3e1353d3dd: Loading layer [==================================================>]
18.99MB/18.99MB
20f1e7953be4: Loading layer [==================================================>]
19.81MB/19.81MB
Loaded image: goharbor/registry-photon:v2.2.1
e359335d9d06: Loading layer [==================================================>]
4.931MB/4.931MB
573c32deac46: Loading layer [==================================================>]
5.926MB/5.926MB
4462384e04f0: Loading layer [==================================================>]
14.86MB/14.86MB
93886c98b389: Loading layer [==================================================>]
27.36MB/27.36MB
481cc53e87f1: Loading layer [==================================================>]
22.02kB/22.02kB
34ddb9fc83e7: Loading layer [==================================================>]
14.86MB/14.86MB
Loaded image: goharbor/notary-server-photon:v2.2.1
f948e4c0caca: Loading layer [==================================================>]
6.783MB/6.783MB
ec1372991658: Loading layer [==================================================>]
9.097MB/9.097MB
1ef3f81e1f85: Loading layer [==================================================>]
1.691MB/1.691MB
Loaded image: goharbor/harbor-portal:v2.2.1
46d871958df2: Loading layer [==================================================>]
8.076MB/8.076MB
03e260326ab5: Loading layer [==================================================>]
3.584kB/3.584kB
6c53b42399ce: Loading layer [==================================================>]
2.56kB/2.56kB
f0859eadaaf8: Loading layer [==================================================>]
53.27MB/53.27MB
48e28227863e: Loading layer [==================================================>]
5.632kB/5.632kB
af9d6bf9cb83: Loading layer [==================================================>]
90.11kB/90.11kB
67d7d6940a94: Loading layer [==================================================>]
11.78kB/11.78kB
07662a79fbff: Loading layer [==================================================>]
54.2MB/54.2MB
6d0fecda12d9: Loading layer [==================================================>]
2.56kB/2.56kB
Loaded image: goharbor/harbor-core:v2.2.1
```

```
324f82f1e2f8: Loading layer [==================================================>]
35.95MB/35.95MB
e13d3998e590: Loading layer [==================================================>]
3.072kB/3.072kB
3735726c1403: Loading layer [==================================================>]
59.9kB/59.9kB
3da48fc3af0e: Loading layer [==================================================>]
61.95kB/61.95kB
Loaded image: goharbor/redis-photon:v2.2.1
6e7fceefe62a: Loading layer [==================================================>]
4.931MB/4.931MB
0148fb852b85: Loading layer [==================================================>]
5.926MB/5.926MB
fcfbd97f83cd: Loading layer [==================================================>]
13.33MB/13.33MB
9d99acddd376: Loading layer [==================================================>]
27.36MB/27.36MB
cb7528f98674: Loading layer [==================================================>]
22.02kB/22.02kB
816b6ef47521: Loading layer [==================================================>]
13.33MB/13.33MB
Loaded image: goharbor/notary-signer-photon:v2.2.1
ece94fe3fa7d: Loading layer [==================================================>]
4.936MB/4.936MB
361117114ba4: Loading layer [==================================================>]
62.71MB/62.71MB
8bcb062f0929: Loading layer [==================================================>]
3.072kB/3.072kB
4486548b56a1: Loading layer [==================================================>]
4.096kB/4.096kB
b3660e86e8c2: Loading layer [==================================================>]
63.53MB/63.53MB
Loaded image: goharbor/chartmuseum-photon:v2.2.1
ad64336d0e51: Loading layer [==================================================>]
77.49MB/77.49MB
3f760c535efc: Loading layer [==================================================>]
54.66MB/54.66MB
ce6390c67a6a: Loading layer [==================================================>]
2.56kB/2.56kB
e56ca8f2c586: Loading layer [==================================================>]
1.536kB/1.536kB
56b738911601: Loading layer [==================================================>]
18.43kB/18.43kB
14c3e8748a68: Loading layer [==================================================>]
4.067MB/4.067MB
5172b1fbd671: Loading layer [==================================================>]
278.5kB/278.5kB
Loaded image: goharbor/prepare:v2.2.1
5d79e0b031e3: Loading layer [==================================================>]
76.08MB/76.08MB
ae7c7f0e9c04: Loading layer [==================================================>]
3.584kB/3.584kB
85ec797b97cb: Loading layer [==================================================>]
3.072kB/3.072kB
0b1fe21c8422: Loading layer [==================================================>]
2.56kB/2.56kB
```

```
9dac10dcafad: Loading layer [==================================================>]
3.072kB/3.072kB
672cf3cb855c: Loading layer [==================================================>]
3.584kB/3.584kB
1fbe5ad20ece: Loading layer [==================================================>]
12.29kB/12.29kB
Loaded image: goharbor/harbor-log:v2.2.1
d4b2501bd60f: Loading layer [==================================================>]
8.076MB/8.076MB
b35ba3bc6760: Loading layer [==================================================>]
17.61MB/17.61MB
aad9bed872f0: Loading layer [==================================================>]
4.608kB/4.608kB
5233fbaf23ed: Loading layer [==================================================>]
18.43MB/18.43MB
Loaded image: goharbor/harbor-exporter:v2.2.1
e92fd18cc2cd: Loading layer [==================================================>]
6.783MB/6.783MB
Loaded image: goharbor/nginx-photon:v2.2.1
43cfa27fb7b9: Loading layer [==================================================>]
63.78MB/63.78MB
381a3761198d: Loading layer [==================================================>]
80.93MB/80.93MB
575a4ef00206: Loading layer [==================================================>]
6.144kB/6.144kB
d283c20b6814: Loading layer [==================================================>]
2.56kB/2.56kB
5ee933fe737a: Loading layer [==================================================>]
2.56kB/2.56kB
f666b92ffe52: Loading layer [==================================================>]
2.56kB/2.56kB
348980754dc5: Loading layer [==================================================>]
2.56kB/2.56kB
ad39d2f7b9b8: Loading layer [==================================================>]
11.26kB/11.26kB
Loaded image: goharbor/harbor-db:v2.2.1
dbebf4744f06: Loading layer [==================================================>]
4.937MB/4.937MB
b8e081520905: Loading layer [==================================================>]
4.096kB/4.096kB
442f06402474: Loading layer [==================================================>]
18.99MB/18.99MB
da1eb793d5c9: Loading layer [==================================================>]
3.072kB/3.072kB
2906b858cfe3: Loading layer [==================================================>]
25.32MB/25.32MB
795547d15c57: Loading layer [==================================================>]
45.14MB/45.14MB
Loaded image: goharbor/harbor-registryctl:v2.2.1


[Step 3]: preparing environment ...

[Step 4]: preparing harbor configs ...
prepare base dir is set to /home/atom/harbor
```

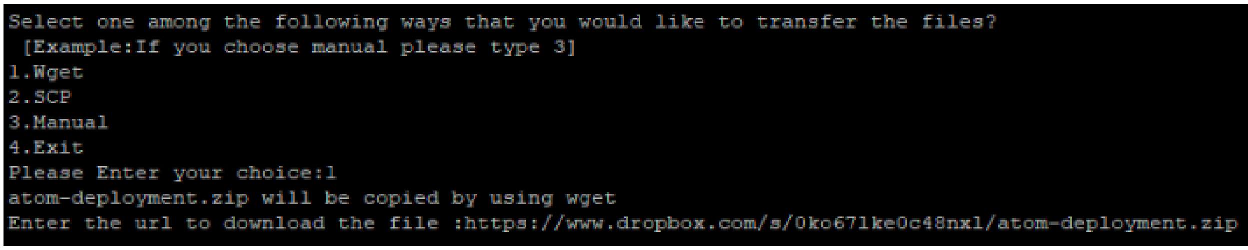
```
WARNING:root:WARNING: HTTP protocol is insecure. Harbor will deprecate http
protocol in the future. Please make sure to upgrade to https
Generated configuration file: /config/portal/nginx.conf
Generated configuration file: /config/log/logrotate.conf
Generated configuration file: /config/log/rsyslog_docker.conf
Generated configuration file: /config/nginx/nginx.conf
Generated configuration file: /config/core/env
Generated configuration file: /config/core/app.conf
Generated configuration file: /config/registry/config.yml
Generated configuration file: /config/registryctl/env
Generated configuration file: /config/registryctl/config.yml
Generated configuration file: /config/db/env
Generated configuration file: /config/jobservice/env
Generated configuration file: /config/jobservice/config.yml
Generated and saved secret to file: /data/secret/keys/secretkey
Successfully called func: create_root_cert
Generated configuration file: /compose_location/docker-compose.yml
Clean up the input dir



[Step 5]: starting Harbor ...
Creating network "harbor_harbor" with the default driver
Creating harbor-log ... done
Creating harbor-portal ... done
Creating redis         ... done
Creating harbor-db     ... done
Creating registryctl   ... done
Creating registry      ... done
Creating harbor-core   ... done
Creating harbor-jobservice ... done
Creating nginx             ... done
✔ ----Harbor has been installed and started successfully.----
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[atom@docker ~]$
```

7. Next we need to download the **Atom deployment.zip** and **images.zip** provided by Anuta Networks team on the docker registry. We can use any of the methods as shown below:



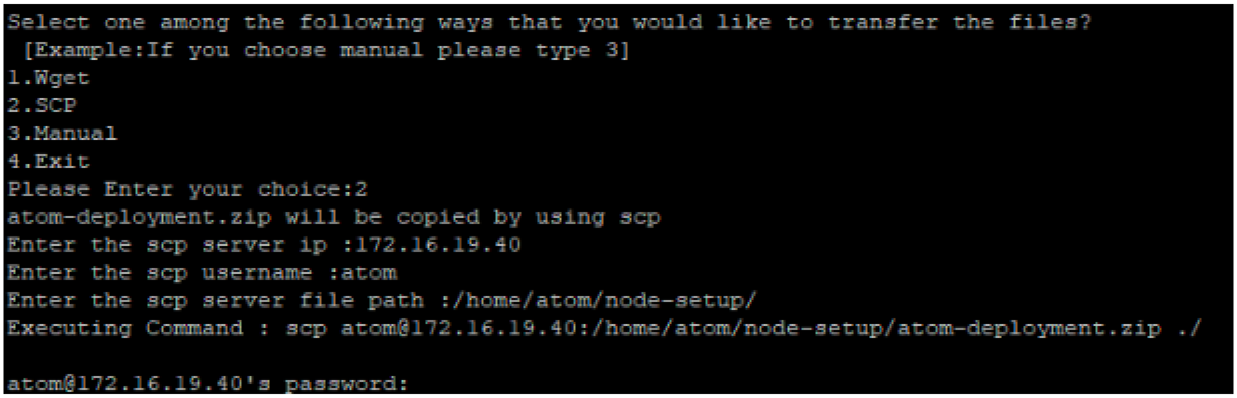
```
Begin Uploading atom-deployment.zip and images.zip

Select one among the following ways that you would like to transfer the atom-deployment zip?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
```

- Wget: utility for non-interactive download of files from the Web. User needs to enter the link as input and the files would be downloaded and extracted automatically.



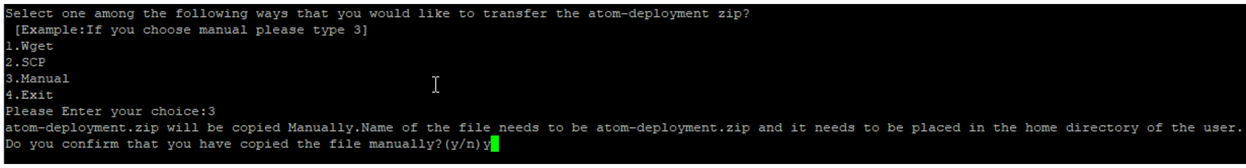
```
Select one among the following ways that you would like to transfer the files?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:1
atom-deployment.zip will be copied by using wget
Enter the url to download the file :https://www.dropbox.com/s/0ko67lke0c48nxl/atom-deployment.zip
```

- SCP: securely transferring computer files between a local host and a remote host based on the Secure Shell (SSH) protocol.



```
Select one among the following ways that you would like to transfer the files?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:2
atom-deployment.zip will be copied by using scp
Enter the scp server ip :172.16.19.40
Enter the scp username :atom
Enter the scp server file path :/home/atom/node-setup/
Executing Command : scp atom@172.16.19.40:/home/atom/node-setup/atom-deployment.zip ./

atom@172.16.19.40's password:
```

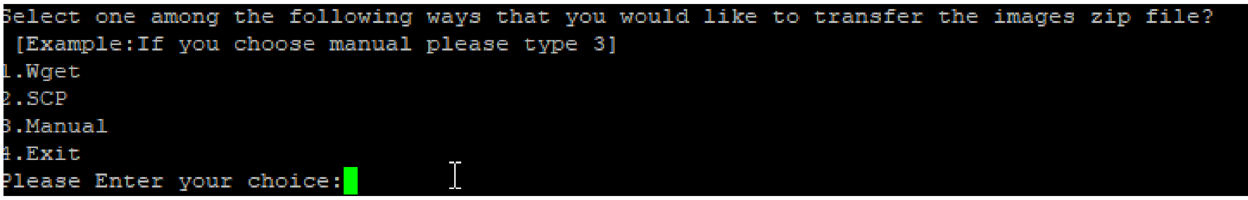
- Manual: User can use any standard file transfer protocol to transfer the files on the home directory of the atom user.



```
Select one among the following ways that you would like to transfer the atom-deployment zip?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:3
atom-deployment.zip will be copied Manually.Name of the file needs to be atom-deployment.zip and it needs to be placed in the home directory of the user.
Do you confirm that you have copied the file manually?(y/n)y
```

  Note : Wget option may not work for offline deployment since we do not have public connectivity .

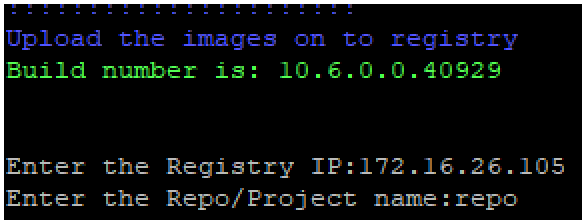
8. To download the **images.zip** provided by Anuta Networks team we can follow the similar procedure as stated above.Please note that the images.zip needs to be copied into the images folder in the home directory.]



```
Select one among the following ways that you would like to transfer the images zip file?
 [Example:If you choose manual please type 3]
1.Wget
2.SCP
3.Manual
4.Exit
Please Enter your choice:
```

   [Troubleshooting: If the images.zip file is not found it is most likely that the folder must have been cleaned at the start of the node_setup script.In this case copy the images.zip again when prompted to do so.]

9. Give registry IP to be used and project name/repo name as provided earlier when setting up the registry.



```
!!!!!!!!!!!!!!!!!!!!!!
Upload the images on to registry
Build number is: 10.6.0.0.40929


Enter the Registry IP:172.16.26.105
Enter the Repo/Project name:repo
```

10. You will observe the following script will execute and output as follows: “sudo python docker-registry.py -r <IP/hostname of host> -p <Repository name given during installation> -t push -v <ATOM BUILD VERSION> ”.

```
Enter the Registry IP:172.16.26.5
Enter the Repo/Project name:release
INFO: Debug logs are sent to atom-registry.log
INFO: ['docker-registry.py', '-r', '172.16.26.5', '-p', 'release', '-t',
'push', '-v', '11.3.0.0.48817']
INFO: Push task is selected
INFO: tar -xf images/databases.tgz
INFO: docker load -qi images/databases/infra-broker:7.0.1.12242021.tar
INFO: docker load -qi images/databases/kafka-operator:0.4.tar
INFO: docker load -qi
images/databases/infra-elasticsearch:7.10.2_with_alerting_reporting_plugi
n.tar
INFO: docker load -qi images/databases/infra-filebeat:770.tar
INFO: docker load -qi
images/databases/infra-distributed-db-webconsole:0.2.tar
INFO: docker load -qi
images/databases/infra-kafka-prometheus-jmx-exporter:0.11.0.tar
INFO: docker load -qi
images/databases/infra-distributed-db-webagent:0.4.tar
INFO: docker load -qi images/databases/postgres-operator:v1.6.3.tar
INFO: docker load -qi images/databases/spilo-13:2.0-p7.tar
INFO: docker load -qi images/databases/infra-logstash:772_150321_v2.tar
INFO: docker load -qi images/databases/infra-zookeeper:7.0.1.12242021.tar
INFO: docker load -qi images/databases/elasticsearch-config:0.5.tar
INFO: docker load -qi images/databases/infra-distributed-db:8.8.7.tar
INFO: docker tag quay.io/release/infra-broker:7.0.1.12242021
172.16.26.5/release/infra-broker:7.0.1.12242021
INFO: docker tag quay.io/release/kafka-operator:0.4
172.16.26.5/release/kafka-operator:0.4
INFO: docker tag
quay.io/release/infra-elasticsearch:7.10.2_with_alerting_reporting_plugin
172.16.26.5/release/infra-elasticsearch:7.10.2_with_alerting_reporting_pl
ugin
INFO: docker tag quay.io/release/infra-filebeat:770
172.16.26.5/release/infra-filebeat:770
INFO: docker tag quay.io/release/infra-distributed-db-webconsole:0.2
172.16.26.5/release/infra-distributed-db-webconsole:0.2
INFO: docker tag
quay.io/release/infra-kafka-prometheus-jmx-exporter:0.11.0
172.16.26.5/release/infra-kafka-prometheus-jmx-exporter:0.11.0
INFO: docker tag quay.io/release/infra-distributed-db-webagent:0.4
172.16.26.5/release/infra-distributed-db-webagent:0.4
```

```
INFO: docker tag
registry.opensource.zalan.do/acid/postgres-operator:v1.6.3
172.16.26.5/release/postgres-operator:v1.6.3
INFO: docker tag registry.opensource.zalan.do/acid/spilo-13:2.0-p7
172.16.26.5/release/spilo-13:2.0-p7
INFO: docker tag quay.io/release/infra-logstash:772_150321_v2
172.16.26.5/release/infra-logstash:772_150321_v2
INFO: docker tag quay.io/release/infra-zookeeper:7.0.1.12242021
172.16.26.5/release/infra-zookeeper:7.0.1.12242021
INFO: docker tag quay.io/release/elasticsearch-config:0.5
172.16.26.5/release/elasticsearch-config:0.5
INFO: docker tag quay.io/release/infra-distributed-db:8.8.7
172.16.26.5/release/infra-distributed-db:8.8.7
INFO: docker push 172.16.26.5/release/infra-broker:7.0.1.12242021
INFO: docker push 172.16.26.5/release/kafka-operator:0.4
INFO: docker push
172.16.26.5/release/infra-elasticsearch:7.10.2_with_alerting_reporting_pl
ugin
INFO: docker push 172.16.26.5/release/infra-filebeat:770
INFO: docker push 172.16.26.5/release/infra-distributed-db-webconsole:0.2
INFO: docker push
172.16.26.5/release/infra-kafka-prometheus-jmx-exporter:0.11.0
INFO: docker push 172.16.26.5/release/infra-distributed-db-webagent:0.4
INFO: docker push 172.16.26.5/release/postgres-operator:v1.6.3
INFO: docker push 172.16.26.5/release/spilo-13:2.0-p7
INFO: docker push 172.16.26.5/release/infra-logstash:772_150321_v2
INFO: docker push 172.16.26.5/release/infra-zookeeper:7.0.1.12242021
INFO: docker push 172.16.26.5/release/elasticsearch-config:0.5
INFO: docker push 172.16.26.5/release/infra-distributed-db:8.8.7
INFO: docker image prune -af
INFO: tar -xf images/linstor.tgz
INFO: docker load -qi images/linstor/csi-provisioner:v3.0.0.tar
INFO: docker load -qi images/linstor/piraeus-ha-controller:v0.2.0.tar
INFO: docker load -qi images/linstor/centos:8.tar
INFO: docker load -qi images/linstor/csi-attacher:v3.3.0.tar
INFO: docker load -qi images/linstor/csi-snapshotter:v4.2.1.tar
INFO: docker load -qi images/linstor/kube-scheduler-amd64:v1.21.9.tar
INFO: docker load -qi images/linstor/livenessprobe:v2.5.0.tar
INFO: docker load -qi images/linstor/drbd9-centos7:v9.1.4.tar
INFO: docker load -qi images/linstor/piraeus-server:v1.17.0.tar
INFO: docker load -qi images/linstor/drbd-reactor:v0.4.4.tar
INFO: docker load -qi images/linstor/etcd:v3.4.15.tar
INFO: docker load -qi images/linstor/piraeus-csi:v0.17.0.tar
INFO: docker load -qi images/linstor/piraeus-operator:v1.7.0.tar
INFO: docker load -qi images/linstor/csi-node-driver-registrar:v2.4.0.tar
INFO: docker load -qi images/linstor/stork:2.6.5.tar
INFO: docker load -qi images/linstor/csi-resizer:v1.3.0.tar
INFO: docker tag k8s.gcr.io/sig-storage/csi-provisioner:v3.0.0
172.16.26.5/release/csi-provisioner:v3.0.0
INFO: docker tag quay.io/piraeusdatastore/piraeus-ha-controller:v0.2.0
172.16.26.5/release/piraeus-ha-controller:v0.2.0
INFO: docker tag quay.io/centos/centos:8 172.16.26.5/release/centos:8
INFO: docker tag k8s.gcr.io/sig-storage/csi-attacher:v3.3.0
172.16.26.5/release/csi-attacher:v3.3.0
INFO: docker tag k8s.gcr.io/sig-storage/csi-snapshotter:v4.2.1
172.16.26.5/release/csi-snapshotter:v4.2.1
```

```
INFO: docker tag k8s.gcr.io/kube-scheduler-amd64:v1.21.9
172.16.26.5/release/kube-scheduler-amd64:v1.21.9
INFO: docker tag k8s.gcr.io/sig-storage/livenessprobe:v2.5.0
172.16.26.5/release/livenessprobe:v2.5.0
INFO: docker tag quay.io/piraeusdatastore/drbd9-centos7:v9.1.4
172.16.26.5/release/drbd9-centos7:v9.1.4
INFO: docker tag quay.io/piraeusdatastore/piraeus-server:v1.17.0
172.16.26.5/release/piraeus-server:v1.17.0
INFO: docker tag quay.io/piraeusdatastore/drbd-reactor:v0.4.4
172.16.26.5/release/drbd-reactor:v0.4.4
INFO: docker tag gcr.io/etcd-development/etcd:v3.4.15
172.16.26.5/release/etcd:v3.4.15
INFO: docker tag quay.io/piraeusdatastore/piraeus-csi:v0.17.0
172.16.26.5/release/piraeus-csi:v0.17.0
INFO: docker tag quay.io/piraeusdatastore/piraeus-operator:v1.7.0
172.16.26.5/release/piraeus-operator:v1.7.0
INFO: docker tag k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.4.0
172.16.26.5/release/csi-node-driver-registrar:v2.4.0
INFO: docker tag quay.io/anuta/stork:2.6.5
172.16.26.5/release/stork:2.6.5
INFO: docker tag k8s.gcr.io/sig-storage/csi-resizer:v1.3.0
172.16.26.5/release/csi-resizer:v1.3.0
INFO: docker push 172.16.26.5/release/csi-provisioner:v3.0.0
INFO: docker push 172.16.26.5/release/piraeus-ha-controller:v0.2.0
INFO: docker push 172.16.26.5/release/centos:8
INFO: docker push 172.16.26.5/release/csi-attacher:v3.3.0
INFO: docker push 172.16.26.5/release/csi-snapshotter:v4.2.1
INFO: docker push 172.16.26.5/release/kube-scheduler-amd64:v1.21.9
INFO: docker push 172.16.26.5/release/livenessprobe:v2.5.0
INFO: docker push 172.16.26.5/release/drbd9-centos7:v9.1.4
INFO: docker push 172.16.26.5/release/piraeus-server:v1.17.0
INFO: docker push 172.16.26.5/release/drbd-reactor:v0.4.4
INFO: docker push 172.16.26.5/release/etcd:v3.4.15
INFO: docker push 172.16.26.5/release/piraeus-csi:v0.17.0
INFO: docker push 172.16.26.5/release/piraeus-operator:v1.7.0
INFO: docker push 172.16.26.5/release/csi-node-driver-registrar:v2.4.0
INFO: docker push 172.16.26.5/release/stork:2.6.5
INFO: docker push 172.16.26.5/release/csi-resizer:v1.3.0
INFO: docker image prune -af
INFO: tar -xf images/kubernetes.tgz
INFO: docker load -qi images/kubernetes/etcd:3.4.13-0.tar
INFO: docker load -qi images/kubernetes/kube-proxy:v1.21.9.tar
INFO: docker load -qi images/kubernetes/coredns:v1.8.0.tar
INFO: docker load -qi images/kubernetes/kube-apiserver:v1.21.9.tar
INFO: docker load -qi images/kubernetes/kube-scheduler:v1.21.9.tar
INFO: docker load -qi
images/kubernetes/kube-controller-manager:v1.21.9.tar
INFO: docker load -qi images/kubernetes/metrics-server:v0.4.2.tar
INFO: docker load -qi images/kubernetes/eventrouter:v0.3.tar
INFO: docker load -qi images/kubernetes/pause:3.4.1.tar
INFO: docker tag k8s.gcr.io/etcd:3.4.13-0
172.16.26.5/release/etcd:3.4.13-0
INFO: docker tag k8s.gcr.io/kube-proxy:v1.21.9
172.16.26.5/release/kube-proxy:v1.21.9
INFO: docker tag k8s.gcr.io/coredns/coredns:v1.8.0
172.16.26.5/release/coredns:v1.8.0
```

```
INFO: docker tag k8s.gcr.io/kube-apiserver:v1.21.9
172.16.26.5/release/kube-apiserver:v1.21.9
INFO: docker tag k8s.gcr.io/kube-scheduler:v1.21.9
172.16.26.5/release/kube-scheduler:v1.21.9
INFO: docker tag k8s.gcr.io/kube-controller-manager:v1.21.9
172.16.26.5/release/kube-controller-manager:v1.21.9
INFO: docker tag k8s.gcr.io/metrics-server/metrics-server:v0.4.2
172.16.26.5/release/metrics-server:v0.4.2
INFO: docker tag gcr.io/heptio-images/eventrouter:v0.3
172.16.26.5/release/eventrouter:v0.3
INFO: docker tag k8s.gcr.io/pause:3.4.1 172.16.26.5/release/pause:3.4.1
INFO: docker push 172.16.26.5/release/etcd:3.4.13-0
INFO: docker push 172.16.26.5/release/kube-proxy:v1.21.9
INFO: docker push 172.16.26.5/release/coredns:v1.8.0
INFO: docker push 172.16.26.5/release/kube-apiserver:v1.21.9
INFO: docker push 172.16.26.5/release/kube-scheduler:v1.21.9
INFO: docker push 172.16.26.5/release/kube-controller-manager:v1.21.9
INFO: docker push 172.16.26.5/release/metrics-server:v0.4.2
INFO: docker push 172.16.26.5/release/eventrouter:v0.3
INFO: docker push 172.16.26.5/release/pause:3.4.1
INFO: docker image prune -af
INFO: tar -xf images/standalone.tgz
INFO: docker load -qi
images/standalone/thanos-receive-controller:latest.tar
INFO: docker load -qi images/standalone/dashboard:v2.0.0-rc5.tar
INFO: docker load -qi images/standalone/modsecurity-spoa:v0.6.tar
INFO: docker load -qi images/standalone/prometheus:v2.30.3.tar
INFO: docker load -qi images/standalone/curator:5.7.6.tar
INFO: docker load -qi images/standalone/postgres_exporter:v0.8.0.tar
INFO: docker load -qi
images/standalone/mc:RELEASE.2020-07-17T02-52-20Z.tar
INFO: docker load -qi images/standalone/jaeger-collector:1.21.0.tar
INFO: docker load -qi images/standalone/haproxy-ingress:v0.11.tar
INFO: docker load -qi images/standalone/thanos:v0.23.1.tar
INFO: docker load -qi images/standalone/rsyslog:latest.tar
INFO: docker load -qi images/standalone/node-exporter:v1.1.2.tar
INFO: docker load -qi images/standalone/spark-dependencies:latest.tar
INFO: docker load -qi images/standalone/pushgateway:v0.8.0.tar
INFO: docker load -qi images/standalone/busybox:1.28.tar
INFO: docker load -qi images/standalone/jaeger-agent:1.21.0.tar
INFO: docker load -qi
images/standalone/jaeger-cassandra-schema:1.21.0.tar
INFO: docker load -qi images/standalone/kafka-manager:latest.tar
INFO: docker load -qi images/standalone/example-hotrod:1.21.0.tar
INFO: docker load -qi images/standalone/metrics-scraper:v1.0.3.tar
INFO: docker load -qi images/standalone/jaeger-query:1.21.0.tar
INFO: docker load -qi images/standalone/alertmanager:v0.21.0.tar
INFO: docker load -qi images/standalone/helm-kubectl-jq:3.1.0.tar
INFO: docker load -qi
images/standalone/minio:RELEASE.2020-07-27T18-37-02Z.tar
INFO: docker load -qi images/standalone/kube-state-metrics:v1.9.8.tar
INFO: docker load -qi images/standalone/cp-schema-registry:7.0.1.tar
INFO: docker load -qi images/standalone/kube-vip:v0.3.8.tar
INFO: docker load -qi images/standalone/burrow-exporter:latest.tar
INFO: docker load -qi images/standalone/echoserver:1.3.tar
INFO: docker load -qi images/standalone/configmap-reload:v0.5.0.tar
```

```
INFO: docker tag quay.io/observatorium/thanos-receive-controller:latest
172.16.26.5/release/thanos-receive-controller:latest
INFO: docker tag quay.io/anuta/dashboard:v2.0.0-rc5
172.16.26.5/release/dashboard:v2.0.0-rc5
INFO: docker tag quay.io/jcmoraisjr/modsecurity-spoa:v0.6
172.16.26.5/release/modsecurity-spoa:v0.6
INFO: docker tag quay.io/prometheus/prometheus:v2.30.3
172.16.26.5/release/prometheus:v2.30.3
INFO: docker tag quay.io/anuta/curator:5.7.6
172.16.26.5/release/curator:5.7.6
INFO: docker tag quay.io/anuta/postgres_exporter:v0.8.0
172.16.26.5/release/postgres_exporter:v0.8.0
INFO: docker tag quay.io/anuta/mc:RELEASE.2020-07-17T02-52-20Z
172.16.26.5/release/mc:RELEASE.2020-07-17T02-52-20Z
INFO: docker tag quay.io/jaegertracing/jaeger-collector:1.21.0
172.16.26.5/release/jaeger-collector:1.21.0
INFO: docker tag quay.io/jcmoraisjr/haproxy-ingress:v0.11
172.16.26.5/release/haproxy-ingress:v0.11
INFO: docker tag quay.io/thanos/thanos:v0.23.1
172.16.26.5/release/thanos:v0.23.1
INFO: docker tag quay.io/anuta/rsyslog:latest
172.16.26.5/release/rsyslog:latest
INFO: docker tag quay.io/prometheus/node-exporter:v1.1.2
172.16.26.5/release/node-exporter:v1.1.2
INFO: docker tag quay.io/jaegertracing/spark-dependencies:latest
172.16.26.5/release/spark-dependencies:latest
INFO: docker tag quay.io/prometheus/pushgateway:v0.8.0
172.16.26.5/release/pushgateway:v0.8.0
INFO: docker tag quay.io/anuta/busybox:1.28
172.16.26.5/release/busybox:1.28
INFO: docker tag quay.io/jaegertracing/jaeger-agent:1.21.0
172.16.26.5/release/jaeger-agent:1.21.0
INFO: docker tag quay.io/jaegertracing/jaeger-cassandra-schema:1.21.0
172.16.26.5/release/jaeger-cassandra-schema:1.21.0
INFO: docker tag quay.io/anuta/kafka-manager:latest
172.16.26.5/release/kafka-manager:latest
INFO: docker tag quay.io/jaegertracing/example-hotrod:1.21.0
172.16.26.5/release/example-hotrod:1.21.0
INFO: docker tag quay.io/anuta/metrics-scraper:v1.0.3
172.16.26.5/release/metrics-scraper:v1.0.3
INFO: docker tag quay.io/jaegertracing/jaeger-query:1.21.0
172.16.26.5/release/jaeger-query:1.21.0
INFO: docker tag quay.io/prometheus/alertmanager:v0.21.0
172.16.26.5/release/alertmanager:v0.21.0
INFO: docker tag quay.io/anuta/helm-kubectl-jq:3.1.0
172.16.26.5/release/helm-kubectl-jq:3.1.0
INFO: docker tag quay.io/anuta/minio:RELEASE.2020-07-27T18-37-02Z
172.16.26.5/release/minio:RELEASE.2020-07-27T18-37-02Z
INFO: docker tag quay.io/coreos/kube-state-metrics:v1.9.8
172.16.26.5/release/kube-state-metrics:v1.9.8
INFO: docker tag quay.io/anuta/cp-schema-registry:7.0.1
172.16.26.5/release/cp-schema-registry:7.0.1
INFO: docker tag ghcr.io/kube-vip/kube-vip:v0.3.8
172.16.26.5/release/kube-vip:v0.3.8
INFO: docker tag quay.io/anuta/burrow-exporter:latest
172.16.26.5/release/burrow-exporter:latest
```

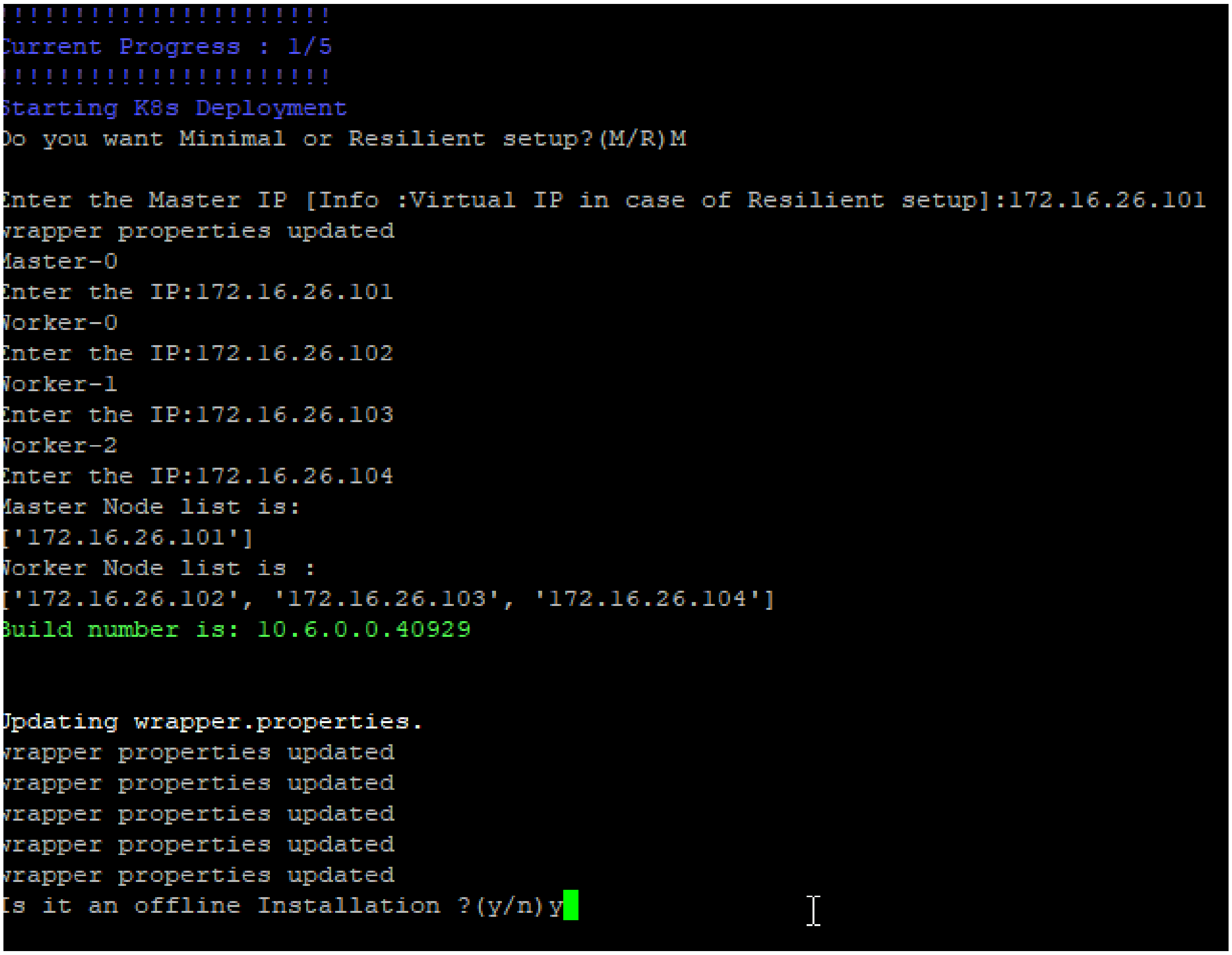
```
INFO: docker tag gcr.io/google_containers/echoserver:1.3
172.16.26.5/release/echoserver:1.3
INFO: docker tag quay.io/anuta/configmap-reload:v0.5.0
172.16.26.5/release/configmap-reload:v0.5.0
INFO: docker push 172.16.26.5/release/thanos-receive-controller:latest
INFO: docker push 172.16.26.5/release/dashboard:v2.0.0-rc5
INFO: docker push 172.16.26.5/release/modsecurity-spoa:v0.6
INFO: docker push 172.16.26.5/release/prometheus:v2.30.3
INFO: docker push 172.16.26.5/release/curator:5.7.6
INFO: docker push 172.16.26.5/release/postgres_exporter:v0.8.0
INFO: docker push 172.16.26.5/release/mc:RELEASE.2020-07-17T02-52-20Z
INFO: docker push 172.16.26.5/release/jaeger-collector:1.21.0
INFO: docker push 172.16.26.5/release/haproxy-ingress:v0.11
INFO: docker push 172.16.26.5/release/thanos:v0.23.1
INFO: docker push 172.16.26.5/release/rsyslog:latest
INFO: docker push 172.16.26.5/release/node-exporter:v1.1.2
INFO: docker push 172.16.26.5/release/spark-dependencies:latest
INFO: docker push 172.16.26.5/release/pushgateway:v0.8.0
INFO: docker push 172.16.26.5/release/busybox:1.28
INFO: docker push 172.16.26.5/release/jaeger-agent:1.21.0
INFO: docker push 172.16.26.5/release/jaeger-cassandra-schema:1.21.0
INFO: docker push 172.16.26.5/release/kafka-manager:latest
INFO: docker push 172.16.26.5/release/example-hotrod:1.21.0
INFO: docker push 172.16.26.5/release/metrics-scraper:v1.0.3
INFO: docker push 172.16.26.5/release/jaeger-query:1.21.0
INFO: docker push 172.16.26.5/release/alertmanager:v0.21.0
INFO: docker push 172.16.26.5/release/helm-kubectl-jq:3.1.0
INFO: docker push 172.16.26.5/release/minio:RELEASE.2020-07-27T18-37-02Z
INFO: docker push 172.16.26.5/release/kube-state-metrics:v1.9.8
INFO: docker push 172.16.26.5/release/cp-schema-registry:7.0.1
INFO: docker push 172.16.26.5/release/kube-vip:v0.3.8
INFO: docker push 172.16.26.5/release/burrow-exporter:latest
INFO: docker push 172.16.26.5/release/echoserver:1.3
INFO: docker push 172.16.26.5/release/configmap-reload:v0.5.0
INFO: docker image prune -af
INFO: tar -xf images/atom.tgz
INFO: docker load -qi images/atom/atom-python3:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-agent-proxy:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-pnp-server:11.3.0.0.48817.tar
INFO: docker load -qi
images/atom/atom-telemetry-engine:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-ml:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-python2:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-workflow-engine:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-isim:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-inventory-mgr:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-core:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-agent:11.3.0.0.48817.tar
INFO: docker load -qi
images/atom/atom-telemetry-exporter:11.3.0.0.48817.tar
INFO: docker load -qi images/atom/atom-scheduler:11.3.0.0.48817.tar
INFO: docker tag quay.io/release/atom-python3:11.3.0.0.48817
172.16.26.5/release/atom-python3:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-agent-proxy:11.3.0.0.48817
172.16.26.5/release/atom-agent-proxy:11.3.0.0.48817
```

```
INFO: docker tag quay.io/release/atom-pnp-server:11.3.0.0.48817
172.16.26.5/release/atom-pnp-server:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-telemetry-engine:11.3.0.0.48817
172.16.26.5/release/atom-telemetry-engine:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-ml:11.3.0.0.48817
172.16.26.5/release/atom-ml:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-python2:11.3.0.0.48817
172.16.26.5/release/atom-python2:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-workflow-engine:11.3.0.0.48817
172.16.26.5/release/atom-workflow-engine:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-isim:11.3.0.0.48817
172.16.26.5/release/atom-isim:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-inventory-mgr:11.3.0.0.48817
172.16.26.5/release/atom-inventory-mgr:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-core:11.3.0.0.48817
172.16.26.5/release/atom-core:11.3.0.0.48817
INFO: docker tag quay.io/atom-agent/atom-agent:11.3.0.0.48817
172.16.26.5/release/atom-agent:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-telemetry-exporter:11.3.0.0.48817
172.16.26.5/release/atom-telemetry-exporter:11.3.0.0.48817
INFO: docker tag quay.io/release/atom-scheduler:11.3.0.0.48817
172.16.26.5/release/atom-scheduler:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-python3:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-agent-proxy:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-pnp-server:11.3.0.0.48817
INFO: docker push
172.16.26.5/release/atom-telemetry-engine:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-ml:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-python2:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-workflow-engine:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-isim:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-inventory-mgr:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-core:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-agent:11.3.0.0.48817
INFO: docker push
172.16.26.5/release/atom-telemetry-exporter:11.3.0.0.48817
INFO: docker push 172.16.26.5/release/atom-scheduler:11.3.0.0.48817
INFO: docker image prune -af
INFO: tar -xf images/calico.tgz
INFO: docker load -qi images/calico/pod2daemon-flexvol:v3.15.5.tar
INFO: docker load -qi images/calico/cni:v3.15.5.tar
INFO: docker load -qi images/calico/node:v3.15.5.tar
INFO: docker load -qi images/calico/kube-controllers:v3.15.5.tar
INFO: docker tag quay.io/calico/pod2daemon-flexvol:v3.15.5
172.16.26.5/release/pod2daemon-flexvol:v3.15.5
INFO: docker tag quay.io/calico/cni:v3.15.5
172.16.26.5/release/cni:v3.15.5
INFO: docker tag quay.io/calico/node:v3.15.5
172.16.26.5/release/node:v3.15.5
INFO: docker tag quay.io/calico/kube-controllers:v3.15.5
172.16.26.5/release/kube-controllers:v3.15.5
INFO: docker push 172.16.26.5/release/pod2daemon-flexvol:v3.15.5
INFO: docker push 172.16.26.5/release/cni:v3.15.5
INFO: docker push 172.16.26.5/release/node:v3.15.5
INFO: docker push 172.16.26.5/release/kube-controllers:v3.15.5
INFO: docker image prune -af
```

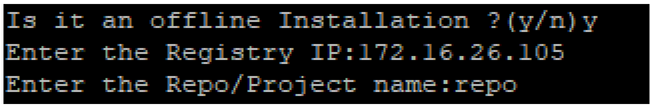
```
INFO: tar -xf images/calico.tgz
INFO: docker load -qi images/calico/pod2daemon-flexvol:v3.15.5.tar
INFO: docker load -qi images/calico/cni:v3.15.5.tar
INFO: docker load -qi images/calico/node:v3.15.5.tar
INFO: docker load -qi images/calico/kube-controllers:v3.15.5.tar
INFO: docker tag quay.io/calico/pod2daemon-flexvol:v3.15.5
172.16.26.5/release/pod2daemon-flexvol:v3.15.5
INFO: docker tag quay.io/calico/cni:v3.15.5
172.16.26.5/release/cni:v3.15.5
INFO: docker tag quay.io/calico/node:v3.15.5
172.16.26.5/release/node:v3.15.5
INFO: docker tag quay.io/calico/kube-controllers:v3.15.5
172.16.26.5/release/kube-controllers:v3.15.5
INFO: docker push 172.16.26.5/release/pod2daemon-flexvol:v3.15.5
INFO: docker push 172.16.26.5/release/cni:v3.15.5
INFO: docker push 172.16.26.5/release/node:v3.15.5
INFO: docker push 172.16.26.5/release/kube-controllers:v3.15.5
INFO: docker image prune -af
INFO: tar -xf images/infra.tgz
INFO: docker load -qi images/infra/infra-grafana:7.5.7.tar
INFO: docker load -qi images/infra/infra-glowroot:0.9.tar
INFO: docker load -qi images/infra/infra-kafka-control:0.2.0.tar
INFO: docker load -qi images/infra/infra-burrow:1.3.8.tar
INFO: docker load -qi images/infra/keycloak:13.1.0.tar
INFO: docker load -qi images/infra/controller:v0.10.2.tar
INFO: docker load -qi images/infra/speaker:v0.10.2.tar
INFO: docker load -qi images/infra/oauth2-proxy:v7.0.1-verifyjwt.tar
INFO: docker load -qi images/infra/infra-kibana:710_opendistro.tar
INFO: docker tag quay.io/release/infra-grafana:7.5.7
172.16.26.5/release/infra-grafana:7.5.7
INFO: docker tag quay.io/release/infra-glowroot:0.9
172.16.26.5/release/infra-glowroot:0.9
INFO: docker tag quay.io/release/infra-kafka-control:0.2.0
172.16.26.5/release/infra-kafka-control:0.2.0
INFO: docker tag quay.io/release/infra-burrow:1.3.8
172.16.26.5/release/infra-burrow:1.3.8
INFO: docker tag quay.io/release/keycloak:13.1.0
172.16.26.5/release/keycloak:13.1.0
INFO: docker tag quay.io/metallb/controller:v0.10.2
172.16.26.5/release/controller:v0.10.2
INFO: docker tag quay.io/metallb/speaker:v0.10.2
172.16.26.5/release/speaker:v0.10.2
INFO: docker tag quay.io/release/oauth2-proxy:v7.0.1-verifyjwt
172.16.26.5/release/oauth2-proxy:v7.0.1-verifyjwt
INFO: docker tag quay.io/release/infra-kibana:710_opendistro
172.16.26.5/release/infra-kibana:710_opendistro
INFO: docker push 172.16.26.5/release/infra-grafana:7.5.7
INFO: docker push 172.16.26.5/release/infra-glowroot:0.9
INFO: docker push 172.16.26.5/release/infra-kafka-control:0.2.0
INFO: docker push 172.16.26.5/release/infra-burrow:1.3.8
INFO: docker push 172.16.26.5/release/keycloak:13.1.0
INFO: docker push 172.16.26.5/release/controller:v0.10.2
INFO: docker push 172.16.26.5/release/speaker:v0.10.2
INFO: docker push 172.16.26.5/release/oauth2-proxy:v7.0.1-verifyjwt
INFO: docker push 172.16.26.5/release/infra-kibana:710_opendistro
INFO: docker image prune -af
```

Once the above script is executed, the docker registry has been installed and setup correctly. We can begin with k8s installation and Atom Installation.

Please follow the steps as stated in section New Kubernetes cluster to bootstrap master and worker nodes and setup the k8s cluster and install Atom.
Note: As it is an offline Installation we do not require internet connection on any of the nodes as long as the registry has been setup properly and NTP server is present to sync the time between all the nodes.
Please note after updating Node IP: **Select yes option for offline installation and provide the registry ip and project/repo name when prompted.**



```
!!!!!!!!!!!!!!!!!!!!!
urrent Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!
tarting K8s Deployment
o you want Minimal or Resilient setup?(M/R)M

nter the Master IP [Info :Virtual IP in case of Resilient setup]:172.16.26.101
rapper properties updated
aster-0
nter the IP:172.16.26.101
orker-0
nter the IP:172.16.26.102
orker-1
nter the IP:172.16.26.103
orker-2
nter the IP:172.16.26.104
aster Node list is:
['172.16.26.101']
orker Node list is :
['172.16.26.102', '172.16.26.103', '172.16.26.104']
uild number is: 10.6.0.0.40929


pdating wrapper.properties.
rapper properties updated
rapper properties updated
rapper properties updated
rapper properties updated
rapper properties updated
s it an offline Installation ?(y/n)y
```



```
Is it an offline Installation ?(y/n)y
Enter the Registry IP:172.16.26.105
Enter the Repo/Project name:repo
```

Atom Installation would be complete and we can proceed by onboarding packages and devices on the platform.

ATOM Remote Agent Deployment

In the ATOM Distributed deployment model, Remote Agent is used to communicate, collect and monitor the networking devices in your infrastructure using standard protocols. Once the agent collects the data, it gets encrypted and sent to Anuta ATOM Server over an outgoing SSL Connection.

The ATOM Agent is an application that runs on a Linux server within your infrastructure as a docker container. ATOM agents have to be installed on each location of your device's infrastructure.

For deployment of ATOM agent across various geographies perform the steps mentioned in the ATOM Remote Agent Deployment Guide [version 10.0]

Procedure of Deploying ATOM in GCP/GKE

ATOM can be deployed on Google Cloud Platform (GCP) Google Kubernetes Engine (GKE) using the “Deployment scripts and files” provided by Anuta.

Prerequisites

11. An Ubuntu/CentOS machine that has access to the internet, so that the deployment scripts can be run. Below are some of the softwares to be installed on that machine.
    a. Helm v3.5.4
        i. Installation procedure: https://helm.sh/docs/intro/install/
    b. Gcloud SDK
        i. Installation procedure: https://cloud.google.com/sdk/docs/install
        ii. Setup the gcloud SDK using “gcloud auth login” and “gcloud auth application-default login” if they are not set.
        iii. Verification can be done using “gcloud container clusters list”
    c. Kubectl installed v1.21
        i. Installation procedure: https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-using-native-package-management
    d. Docker installed v20.10 and above
        i. Installation procedure: https://docs.docker.com/engine/install/

   e. Python2.7 pip package
      i. Install python-pip using “sudo apt-get install python-pip” or “sudo yum install python-pip” depending on the distro being used.
      ii. Install the following packages using the below command
         1. sudo pip install pyyaml==3.13
         2. sudo pip install requests==2.20.0
         3. sudo pip install setuptools==40.5.0
         4. sudo pip install cryptography==2.3.1
         5. sudo pip install pyJWT==1.6.4
         6. sudo pip install cachetools==2.1.0
         7. sudo pip install kubernetes==11.0.0

12. A site-to-site VPN setup between your datacenter and GCP created for ATOM to reach devices.
13. If the linux machine is created on the GCP, then confirm that the service account mapped has enough privileges to run as sudo. The permissions for the service account to have Kubernetes Engine, Compute Engine and Compute OS privileges to ensure cluster role creation is allowed during ATOM installation.

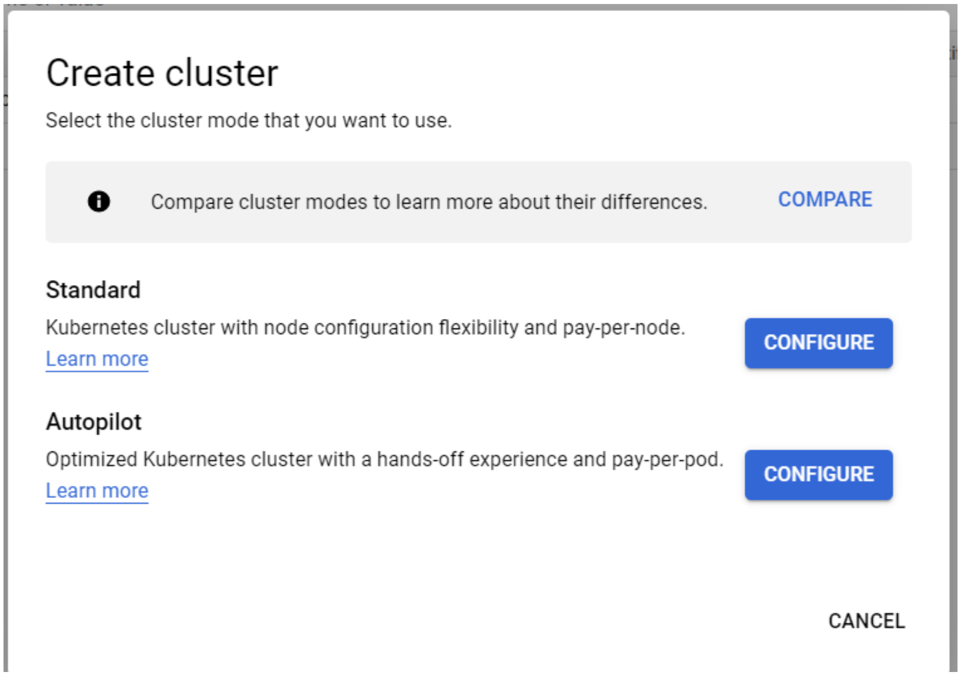
Deploying New K8s Cluster

After ensuring that the prerequisites are taken care, perform the following steps:

Minimal Setup:

Login to your GCP console and navigate to the Kubernetes Engine tab.

- ❏ Click on the Create button which would open the option of cluster models
- ❏ Select the configure button of Standard kind



Create cluster

Select the cluster mode that you want to use.

Compare cluster modes to learn more about their differences.

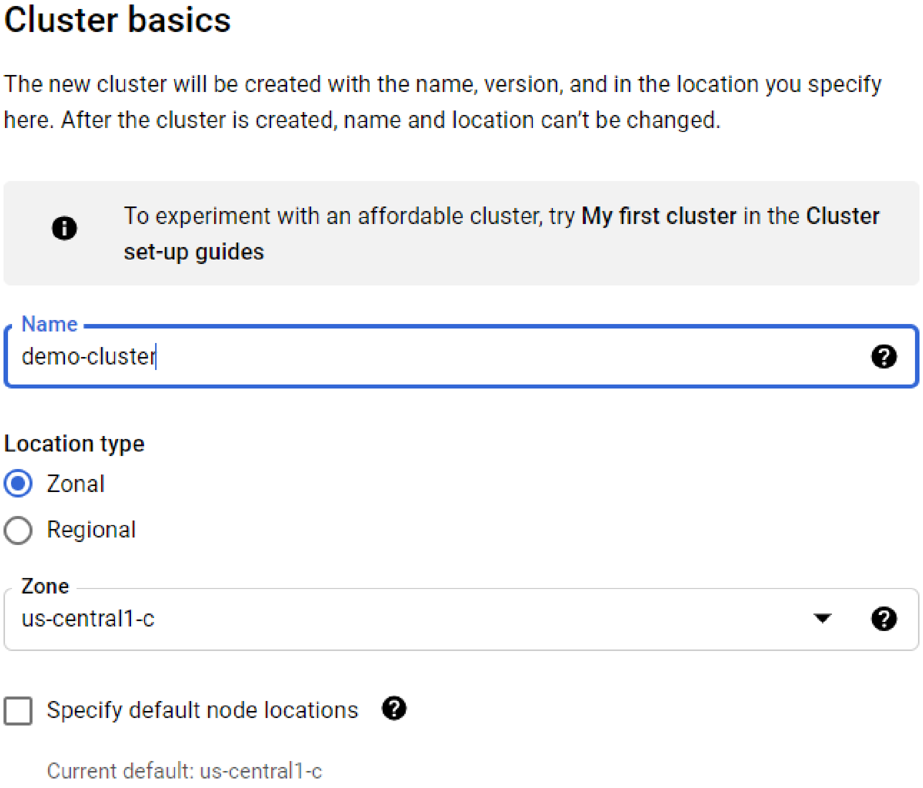
Standard

Kubernetes cluster with node configuration flexibility and pay-per-node.

Autopilot

Optimized Kubernetes cluster with a hands-off experience and pay-per-pod.

❑ Provide the name of the cluster and select the location of choice. Location to be set to Zonal and choice of the zone can be selected from the list available.



Cluster basics

The new cluster will be created with the name, version, and in the location you specify here. After the cluster is created, name and location can't be changed.

To experiment with an affordable cluster, try **My first cluster** in the **Cluster set-up guides**

Name: demo-cluster

Location type

● Zonal

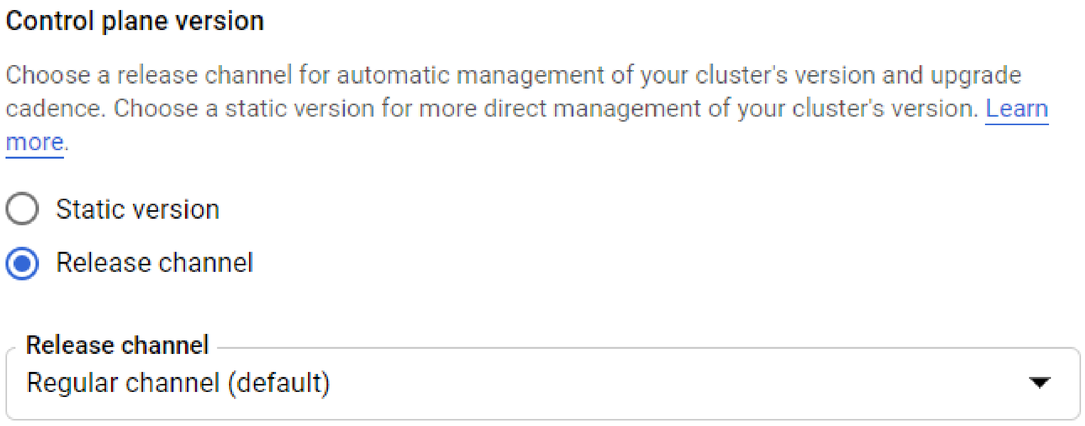
○ Regional

Zone: us-central1-c

□ Specify default node locations

Current default: us-central1-c

❑ Proceed to control plane version selection and select the Release channel radio button. Select the Regular channel from the list and ensure that the kubernetes version falls into v1.21



Control plane version

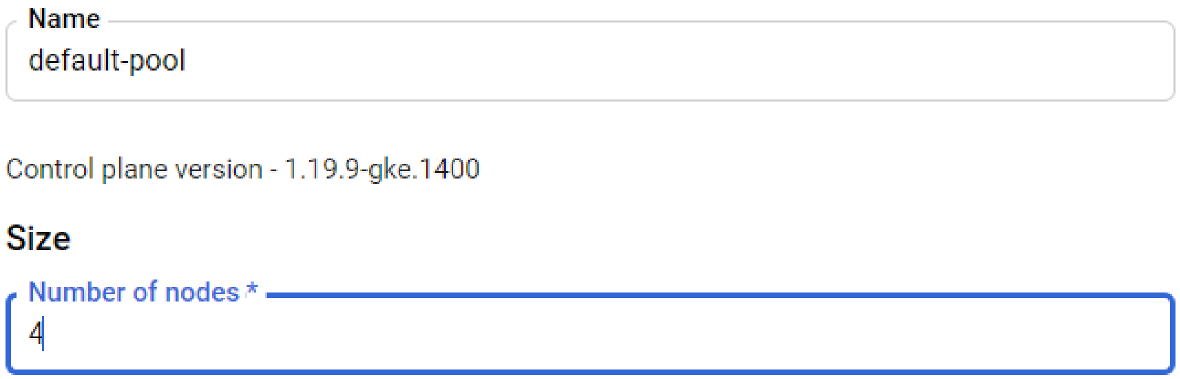
Choose a release channel for automatic management of your cluster's version and upgrade cadence. Choose a static version for more direct management of your cluster's version. Learn more.

○ Static version

● Release channel

Release channel: Regular channel (default)

❑ At the left pane, select the default node pool, provide the name and number of nodes to 4(as per minimal deployment size)



Name: default-pool

Control plane version - 1.19.9-gke.1400

Size

Number of nodes *: 4

❑ Proceed to the nodes section, select the node type as “*Ubuntu with Docker(ubuntu)*”. Set the size to “*e2-highmem-4*” under E2 series.

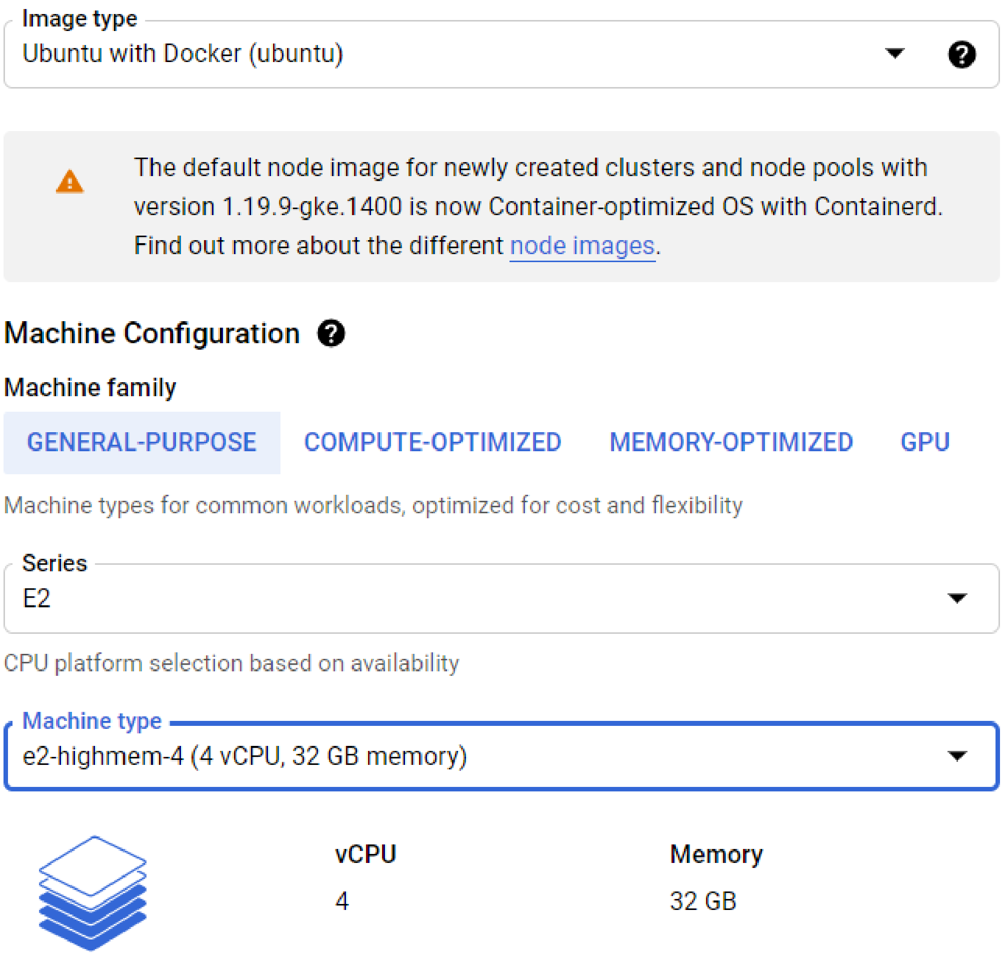


Image type
Ubuntu with Docker (ubuntu)

The default node image for newly created clusters and node pools with version 1.19.9-gke.1400 is now Container-optimized OS with Containerd. Find out more about the different node images.

Machine Configuration

Machine family

Machine types for common workloads, optimized for cost and flexibility

Series
E2

CPU platform selection based on availability

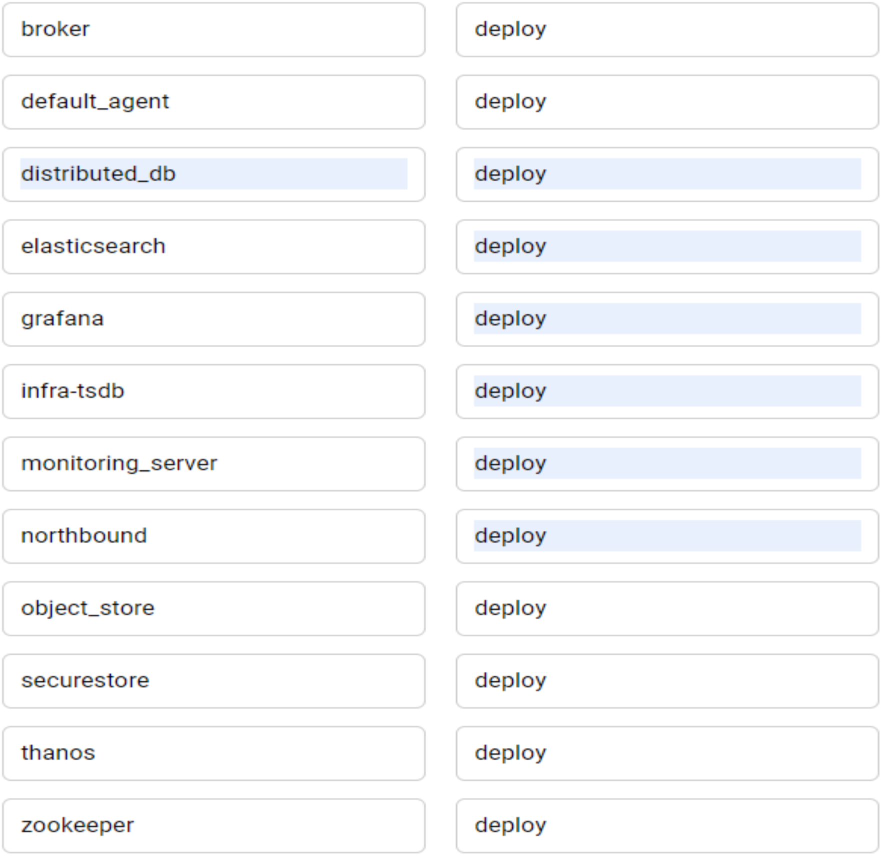
Machine type
e2-highmem-4 (4 vCPU, 32 GB memory)

vCPU	Memory
4	32 GB

❑ Proceed to the Security tab under nodepool, and select the service account created for the same having enough privileges to host the compute instance.

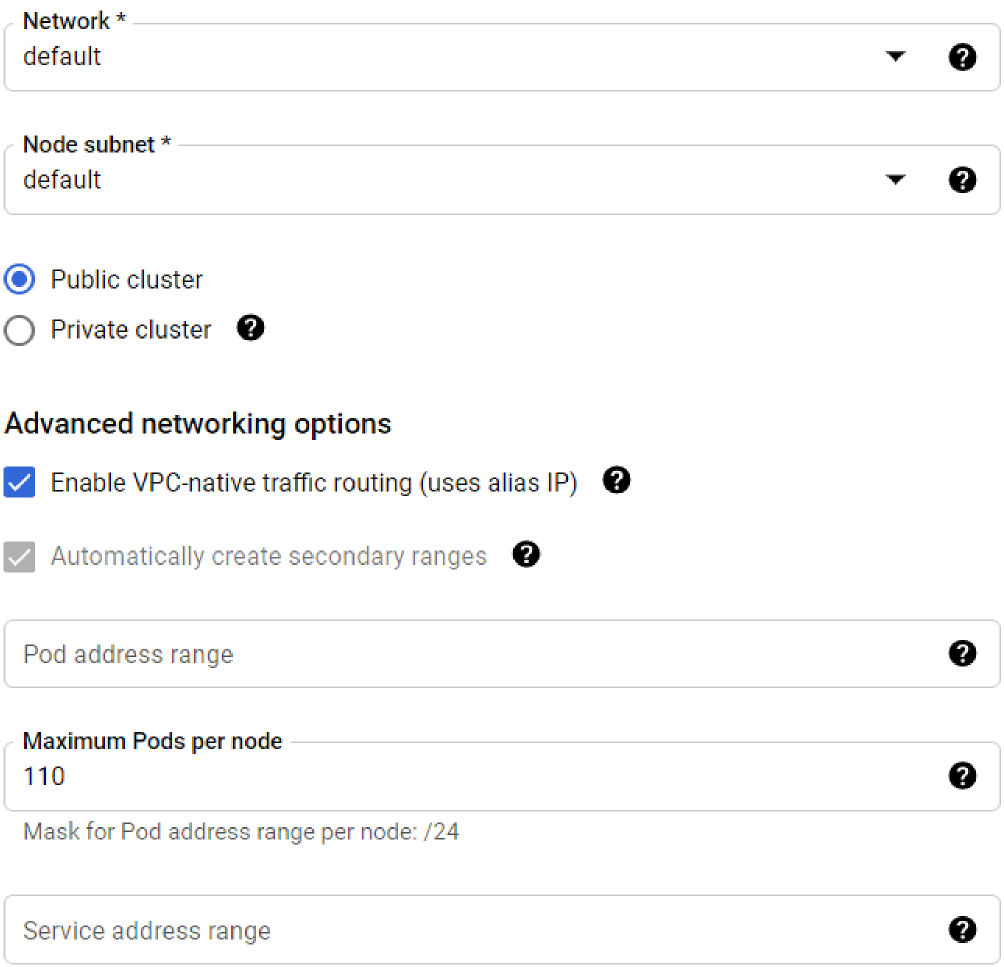
❑ Proceed to the Metadata tab under nodepool, provide the following kubernetes labels

```
broker=deploy
default_agent=deploy
distributed_db=deploy
elasticsearch=deploy
grafana=deploy
infra-tsdb=deploy
monitoring_server=deploy
northbound=deploy
object_store=deploy
securestore=deploy
thanos=deploy
zookeeper=deploy
```



broker	deploy
default_agent	deploy
distributed_db	deploy
elasticsearch	deploy
grafana	deploy
infra-tsdb	deploy
monitoring_server	deploy
northbound	deploy
object_store	deploy
securestore	deploy
thanos	deploy
zookeeper	deploy

- Proceed to the Networking tab, select the network and node subnet as per the lab networking done.
- Select the Public or Private cluster depending on the choice. Provide the POD CIDR and Service CIDR if there are any which accordingly adds to the kubernetes pods.



Network *
default

Node subnet *
default

● Public cluster
○ Private cluster

Advanced networking options

☑ Enable VPC-native traffic routing (uses alias IP)

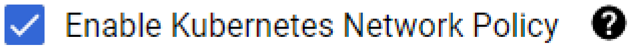
☑ Automatically create secondary ranges

Pod address range

Maximum Pods per node
110
Mask for Pod address range per node: /24

Service address range

- ❑ Set the “Enable Kubernetes Network Policy” by selecting the checkbox. Optionally select if other options are required.



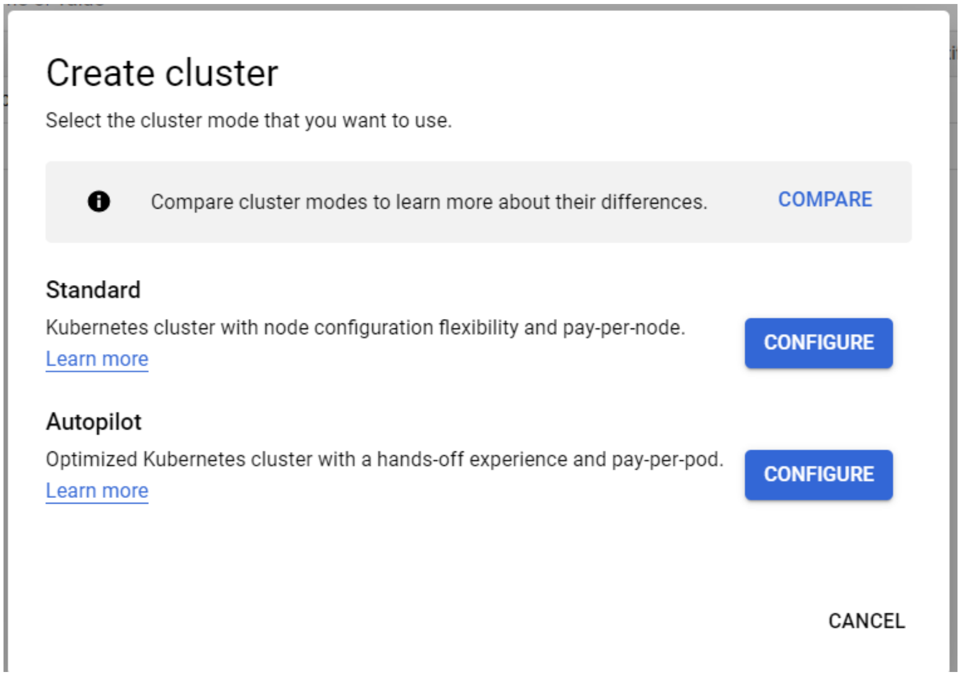
☑ Enable Kubernetes Network Policy

- ❑ Optionally set if there are any options required at Security, Metadata and Features tab as required.
- ❑ Finally select Create at the bottom to create the kubernetes cluster on GKE.

Resilient-HA Setup:

Login to your GCP console and navigate to the Kubernetes Engine tab.

- ❑ Click on the Create button which would open the option of cluster models
- ❑ Select the configure button of Standard kind



Create cluster

Select the cluster mode that you want to use.

Compare cluster modes to learn more about their differences. COMPARE

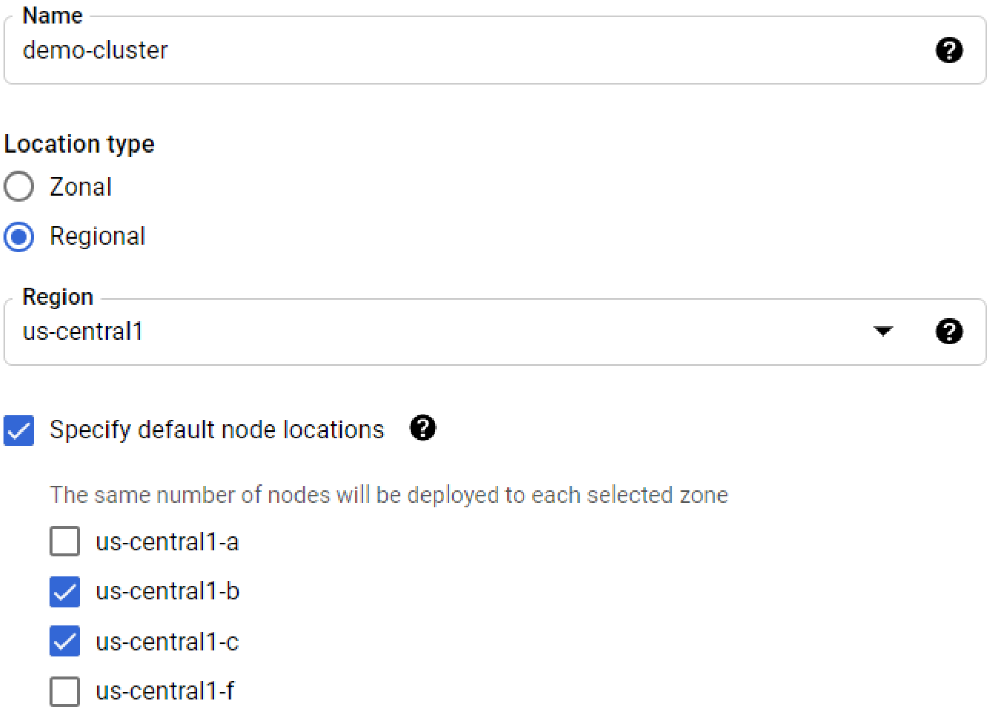
Standard

Kubernetes cluster with node configuration flexibility and pay-per-node.
Learn more

Autopilot

Optimized Kubernetes cluster with a hands-off experience and pay-per-pod.
Learn more

❑ Provide the name of the cluster and select the location of choice. Location to be set to Regional and choice of the zones(2 to 3 zones) can be selected from the list available.



Name: demo-cluster

Location type

○ Zonal

● Regional

Region: us-central1

☑ Specify default node locations

The same number of nodes will be deployed to each selected zone

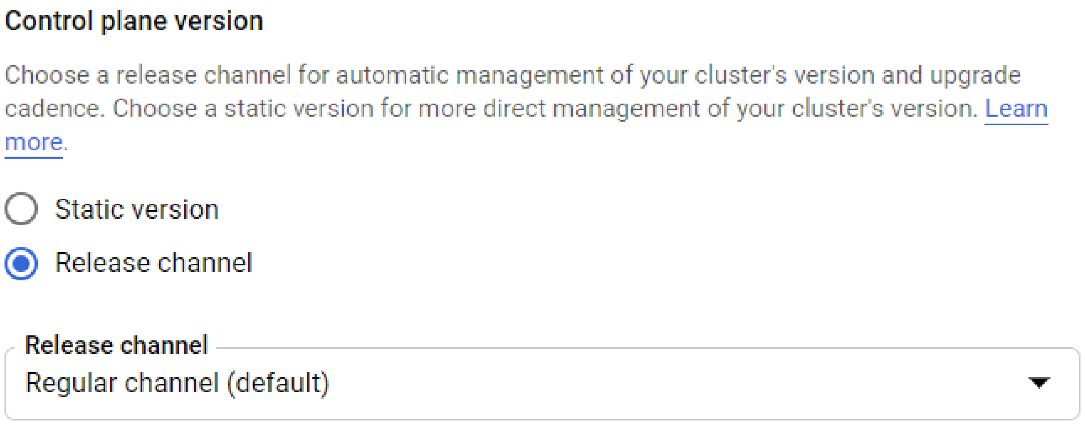
□ us-central1-a

☑ us-central1-b

☑ us-central1-c

□ us-central1-f

❑ Proceed to control plane version selection and select the Release channel radio button. Select the Regular channel from the list and ensure that the kubernetes version falls into v1.21



Control plane version

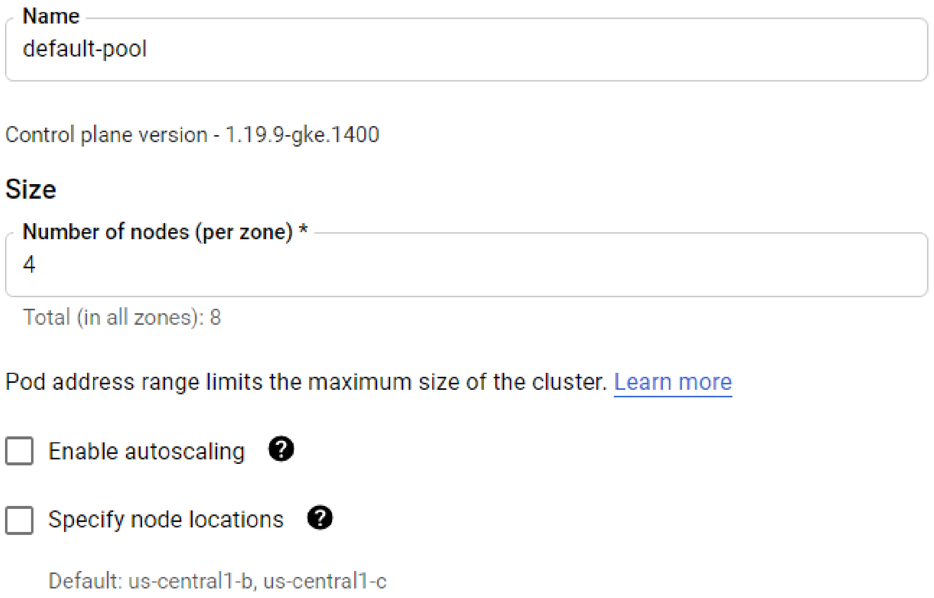
Choose a release channel for automatic management of your cluster's version and upgrade cadence. Choose a static version for more direct management of your cluster's version. Learn more.

○ Static version

● Release channel

Release channel: Regular channel (default)

❑ At the left pane, select the default node pool, provide the name and number of nodes to 4 so the total number of nodes is 8(as per resilient deployment size requirement). If the number of zones selected are 3 then provide the number of nodes to 3 per zone so the total number of nodes is 9.



Name: default-pool

Control plane version - 1.19.9-gke.1400

Size

Number of nodes (per zone) *: 4

Total (in all zones): 8

Pod address range limits the maximum size of the cluster. Learn more

□ Enable autoscaling

□ Specify node locations

Default: us-central1-b, us-central1-c

Note: If node pools need to be separated across zones, create multiple node pool and select the specific node locations of the choice.

❑ Proceed to the nodes section, select the node type as "*Ubuntu with Docker(ubuntu)*". Set the size to "*e2-highmem-4*" under E2 series.

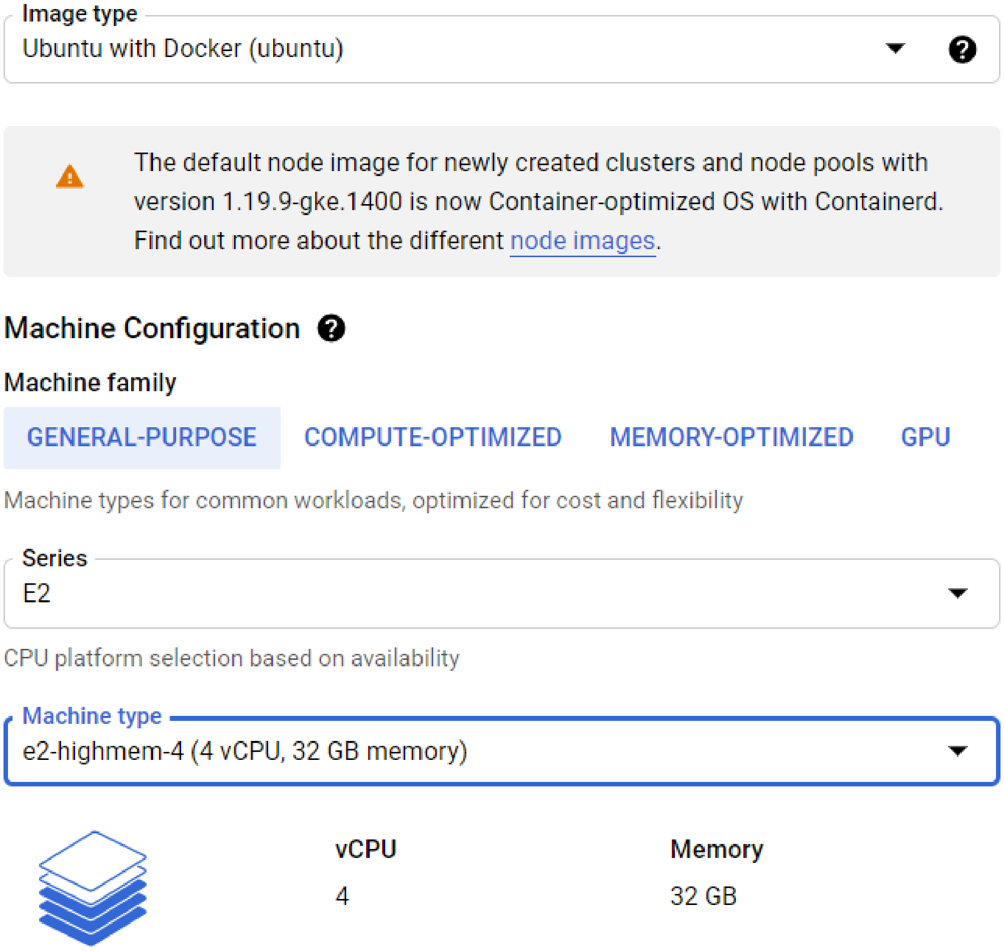


Image type
Ubuntu with Docker (ubuntu)

The default node image for newly created clusters and node pools with version 1.19.9-gke.1400 is now Container-optimized OS with Containerd. Find out more about the different node images.

Machine Configuration

Machine family

Machine types for common workloads, optimized for cost and flexibility

Series
E2

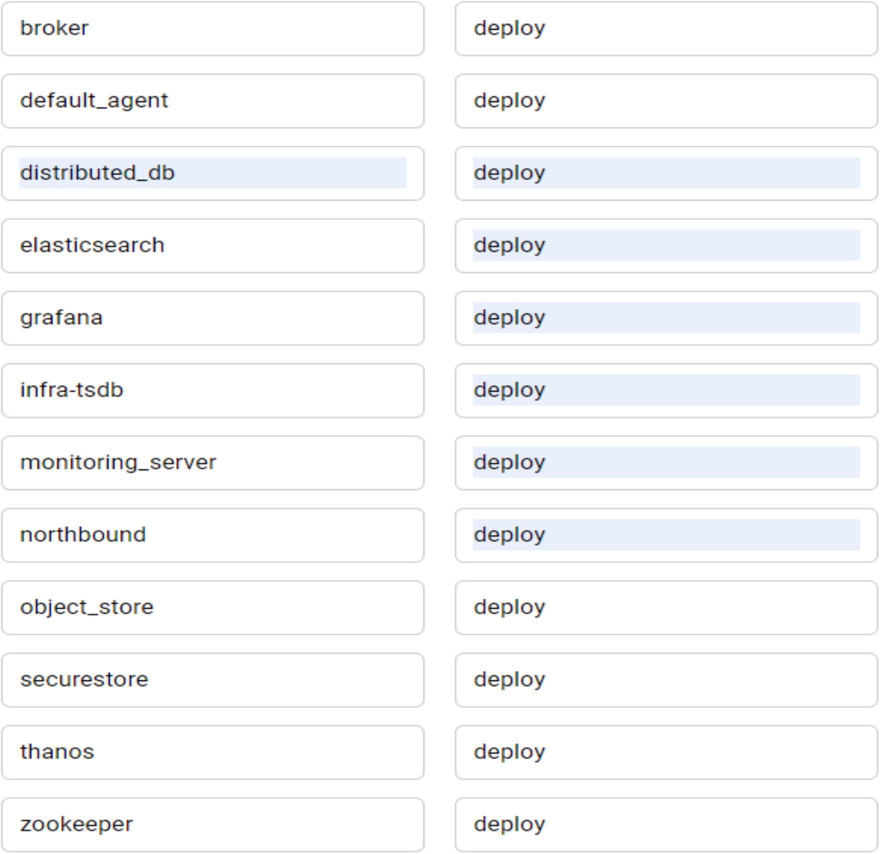
CPU platform selection based on availability

Machine type
e2-highmem-4 (4 vCPU, 32 GB memory)

vCPU	Memory
4	32 GB

- ❑ Proceed to the Security tab under nodepool, and select the service account created for the same having enough privileges to host the compute instance.
- ❑ Proceed to the Metadata tab under nodepool, provide the following kubernetes labels

```
broker=deploy

default_agent=deploy

distributed_db=deploy

elasticsearch=deploy

grafana=deploy

infra-tsdb=deploy

monitoring_server=deploy

northbound=deploy

object_store=deploy

securestore=deploy

thanos=deploy

zookeeper=deploy
```

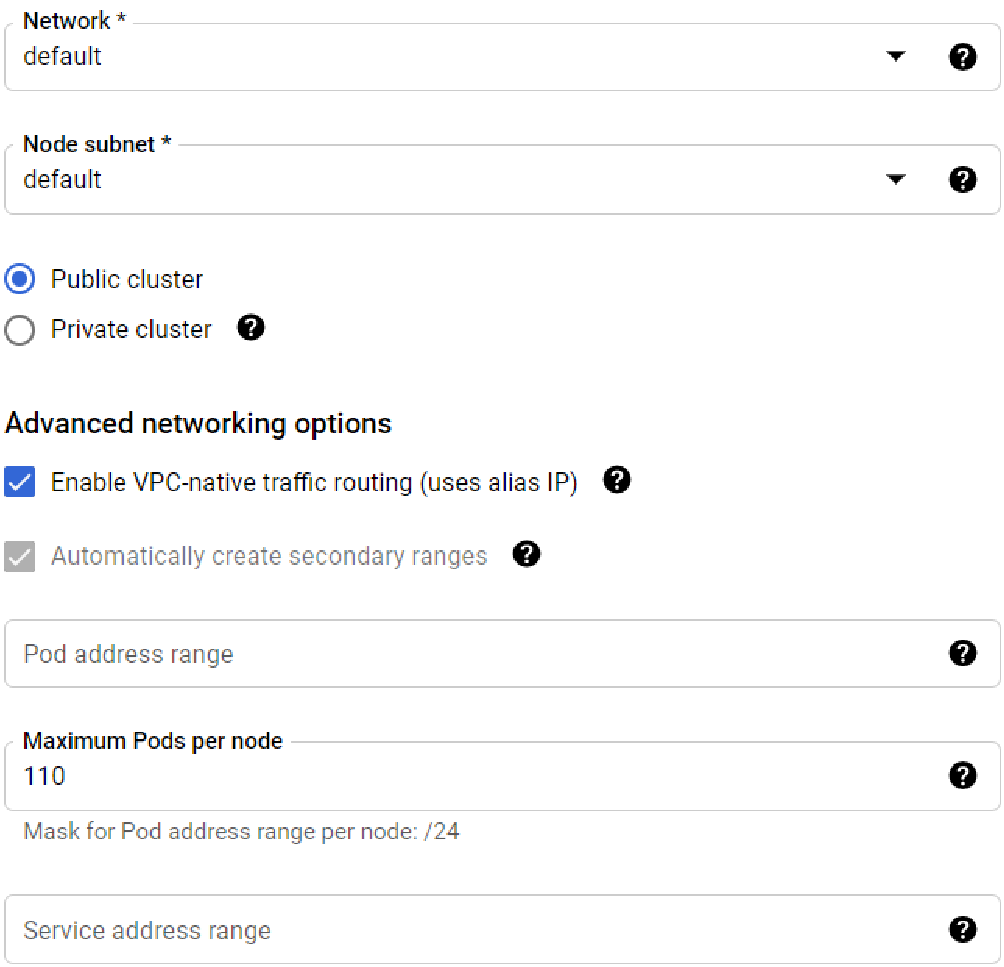


broker	deploy
default_agent	deploy
distributed_db	deploy
elasticsearch	deploy
grafana	deploy
infra-tsdb	deploy
monitoring_server	deploy
northbound	deploy
object_store	deploy
securestore	deploy
thanos	deploy
zookeeper	deploy

Note: If multiple node pools are set, above node labels have to be provided for all the node pools.

❑ Proceed to the Networking tab, select the network and node subnet as per the lab networking done.

❑ Select the Public or Private cluster depending on the choice. Provide the POD CIDR and Service CIDR if there are any which accordingly adds to the kubernetes pods.



Network *
default

Node subnet *
default

● Public cluster
○ Private cluster

Advanced networking options

☑ Enable VPC-native traffic routing (uses alias IP)

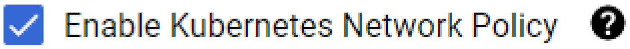
☑ Automatically create secondary ranges

Pod address range

Maximum Pods per node
110
Mask for Pod address range per node: /24

Service address range

❑ Set the “Enable Kubernetes Network Policy” by selecting the checkbox. Optionally select if other options are required.



☑ Enable Kubernetes Network Policy

❑ Optionally set if there are any options required at Security, Metadata and Features tab as required.
❑ Finally select Create at the bottom to create the kubernetes cluster on GKE.

Deploying ATOM

After ensuring that the prerequisites described in the section “Prerequisites” are taken care of, perform the following steps:

1. Login to your linux machine and connect to cluster

```
# gcloud container clusters get-credentials <CLUSTER NAME> --region
<REGION> --project <PROJECT NAME>

# sudo gcloud container clusters get-credentials <CLUSTER NAME>
--region <REGION> --project <PROJECT NAME>

Example: gcloud container clusters get-credentials demo-cluster
--region us-central1 --project anuta-atom-gke
```

Note: running sudo is required since deployment scripts run with sudo

2. Unzip the deployment-scripts folder, provided by Anuta, described in the section, “Deployment scripts and files”. Update wrapper.properties file accordingly.
   a. Cross verify if build number is set
   b. Cross verify if deployment_type is set to **“gcloud”**
   c. Set the public key to “enable” or “disable” depending on cluster type. When enabled, LoadBalancers created for ATOM would have public access over the internet.
   d. Update size to required value like “minimal” or “resilient”
   e. Set the zonal_resiliency to “enable” if the size is resilient to spread workloads across zones. For minimal size, this can be set to disable.
   f. Cross verify if image_pull is set to "quay"
   g. Cross verify if organization is set to "release"

3. Deploy ATOM by executing the following script

```
# sudo python deploy_atom.py
```

4. Executing above steps will complete the ATOM deployment.
   After deployment is completed, the URL’s to access the ATOM application can be fetched by running

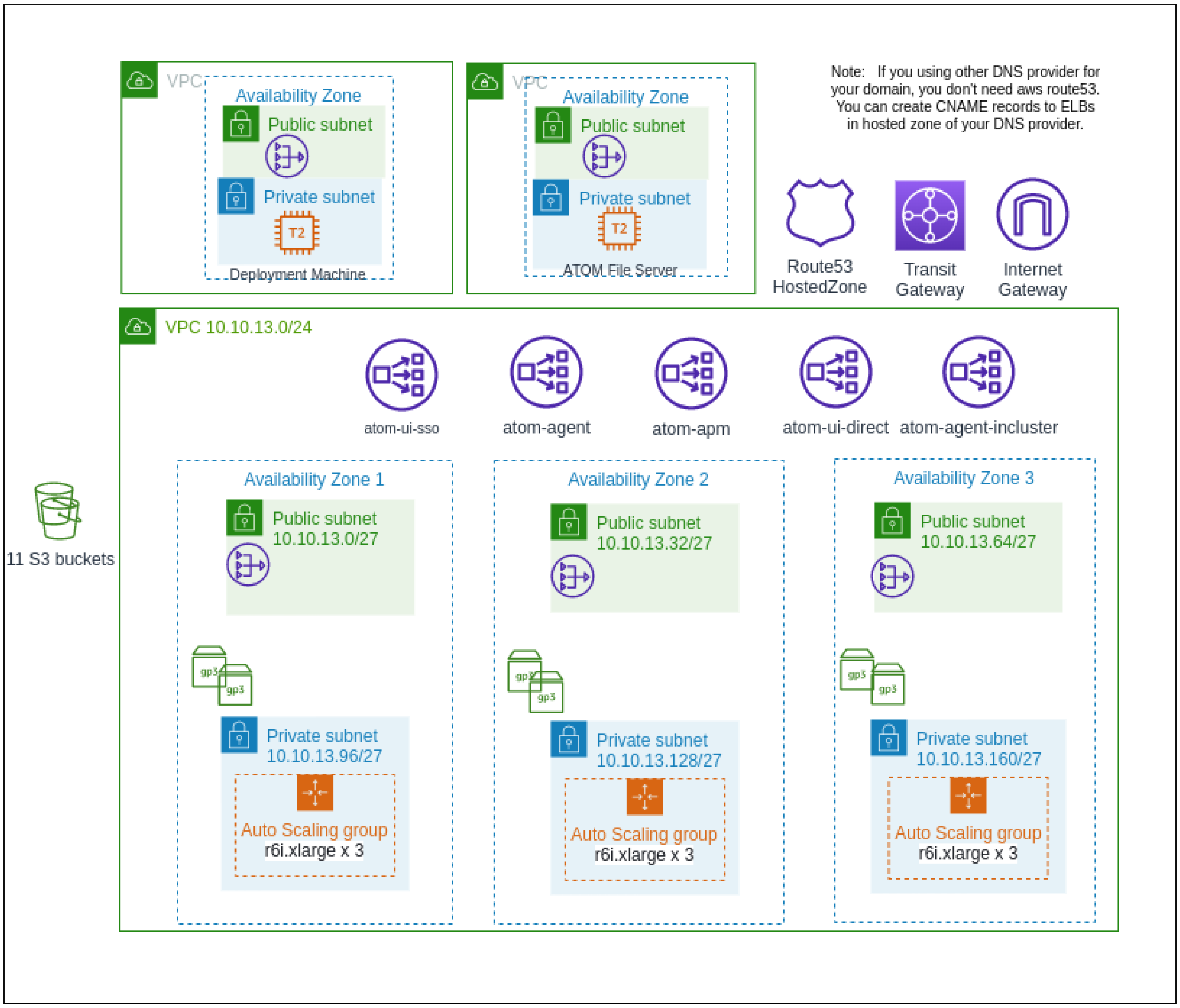
```
# kubectl get svc -n atom
```

   For local deployments, the services are accessible via nodePorts and for cloud deployments the services are accessible via LoadBalancers.

Procedure of Deploying ATOM in AWS

ATOM can be deployed on Amazon Web Services (AWS) Elastic Kubernetes Service (EKS) using the “Deployment scripts and files” provided by Anuta.

Following diagram depicts ATOM deployment in terms of AWS resources.



Note: If you using other DNS provider for your domain, you don't need aws route53. You can create CNAME records to ELBs in hosted zone of your DNS provider.

Prerequisites

1. Deployment Machine - An Ubuntu machine running with v18.04 that has access to the internet, so that the deployment scripts can be run.
   Spec:
   cpu: 2 vcpu
   mem: 4GB
   storage: 50GB
   Image: Ubuntu - 18.04.06 LTS (Bionic Beaver)
   OR
   aws instance type: t2-medium
   Community AMI: ubuntu-bionic-18.04-amd64-server-20211129 - ami-074251216af698218
   SecurityGroup/FW Rules:
   a. Allow inbound ssh access to the ubuntu machine from DC.
   b. Allow all outbound to the internet.
   Below software must be installed on that machine.
   a. Helm v3.5.4

i. wget -q https://get.helm.sh/helm-v3.5.4-linux-amd64.tar.gz

ii. tar -zxvf helm-v3.5.4-linux-amd64.tar.gz

iii. sudo mv linux-amd64/helm /usr/local/bin/

b. AWS CLI v2

i. Installation procedure: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install

c. EKSCTL 0.79.0

i. Installation procedure: https://github.com/weaveworks/eksctl#installation

d. Kubectl installed v1.21 and above

i. Installation procedure: https://v1-21.docs.kubernetes.io/docs/tasks/tools/install-kubectl-linux/

e. python 2.7

i. sudo apt install python

f. Kubernetes pip package v11.0.0

i. Install python-pip using “sudo apt-get install python-pip”

ii. Upgrade the pip to latest using “sudo pip install pip --upgrade”

iii. Installed using “sudo *pip install kubernetes==11.0.0*”

g. Paramiko package v2.6.0

i. Installed using “sudo *pip install paramiko*”

Reboot the system *“sudo reboot”*

2. Connectivity between on-premises DC and AWS VPCs. Please refer to section “AWS connectivity options” in Appendix for more information.
3. A /24 ip block as vpc cidr - ATOM deployment scripts use eksctl tool to create a kubernetes cluster on AWS. eksct by default creates a dedicated vpc and creates a eks cluster in the vpc. Please choose a vpc cidr to accommodate all the future needs considering the cluster expansion if needed.
   Please refer to https://eksctl.io/usage/vpc-networking/ for more information about cidr planning.
4. ELBs

a. All the ELBs are created by kubernetes service manifests.

b. All ELBs are of type NLB and default internal. If you have a requirement to deploy “external” ELB, please modify values in wrapper.properties file - refer to section Deploying ATOM.

c. ATOM by default uses self signed certificates. If you prefer to use the proper certificates, please refer to the section Custom SSL Certificate for ATOM

5. User with following IAM privileges are required

a. IAM User with following permissions to create EKS cluster https://eksctl.io/usage/minimum-iam-policies/.,VPC full access policy.

b. IAM User with s3 Full access policy. This is needed for storing device configuration in s3 using minio

c. Create policy to provision volumes with type gp3.
i. From AWS console select Resources ⟶ IAM ⟶ Access Management ⟶ Policies

.

Click on create policy and select JSON and paste following json config.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ec2:AttachVolume",
            "ec2:CreateSnapshot",
            "ec2:CreateTags",
            "ec2:CreateVolume",
            "ec2:DeleteSnapshot",
            "ec2:DeleteTags",
            "ec2:DeleteVolume",
            "ec2:DescribeInstances",
            "ec2:DescribeSnapshots",
            "ec2:DescribeTags",
            "ec2:DescribeVolumes",
            "ec2:DetachVolume"
         ],
         "Resource": "*"
      }
   ]
}
```

ii. Attach created policy to nodegroup arn used by cluster nodes.
From AWS console select Resources ⟶ IAM ⟶ Access Management ⟶ Roles.
Find the nodegroup-arn (identified by cluster-name) used by nodes and click on that role. And **select permissions** and click on **Attach policies** and add the above policy to the role.

6. File server - An optional component needed only if server ZTP provisioning is used. Please refer to section File Server for ATOM ZTP in AWS.

Deploying New K8s Cluster

After ensuring that the prerequisites are taken care, perform the following steps:

❑ Login to the Ubuntu machine and follow below steps to configure aws cli
aws configure

❑ Execute below command which does Kubernetes cluster addition with prefered node-type as *r6i.xlarge*

eksctl create cluster --name {cluster-name} --version 1.21 --region ***<region-name>***
--node-type r6i.xlarge --without-nodegroup --node-volume-size 50
--node-private-networking --vpc-cidr ***<vpc-cidr>*** *--ssh-access=true*
--ssh-public-key={access-key} --vpc-nat-mode HighlyAvailable

Ex: eksctl create cluster --name aws-deploy-test --version 1.21 --region us-west-2
--node-type r6i.xlarge --without-nodegroup --node-volume-size 50
--node-private-networking --vpc-cidr 192.168.66.0/24 --ssh-access=true
--ssh-public-key=atom --vpc-nat-mode HighlyAvailable

NOTE*:* If ssh public key is missing then generate using ssh-keygen
If want to deploy on specific zones in a region *"--zones=us-west-2a,us-west-2b"*

❑ Delete the Amazon VPC CNI
kubectl delete ds aws-node -n kube-system

❑ Install Calico
Unzip the deployment-scripts folder, provided by Anuta, described in the section, "Deployment scripts and files". From the atom-deployment folder, go into scripts using *"cd scripts*" and run below command.

kubectl apply -f calico.yaml

❑ Now we have a new VPC created for the EKS cluster. At this point please make sure the connectivity between the deployment machine and eks cluster VPC. Please refer to section "AWS connectivity options" in Appendix for more information.

❑ Create a nodegroup for worker nodes.
Based on type of setup being created, execute respective command

Minimal Setup:
eksctl create nodegroup --name {nodegroup-name} --nodes {total-number-of-nodes} --nodes-min {minimum-number-of-nodes-in-availability-zone} --nodes-max {maximum-number-of-nodes-availability-zone} --node-volume-size 50 --cluster {cluster-name} --node-private-networking --node-type r6i.xlarge --ssh-access --node-labels={labels} --managed

Example:

```
eksctl create nodegroup --name testing-nodegroup --nodes 4 --nodes-min 4 --nodes-max
4 --node-volume-size 50 --cluster $1  --node-private-networking --node-type
r6i.xlarge
--node-labels=zookeeper=deploy,broker=deploy,grafana=deploy,elasticsearch=deploy,obje
```

```
ct_store=deploy,distributed_db=deploy,default_agent=deploy,securestore=deploy,monitor
ing_server=deploy,infra-tsdb=deploy,thanos=deploy,agent1=deploy,northbound=deploy
--ssh-access --managed
```

Resilient-HA Setup:

eksctl create nodegroup --name {nodegroup-name} --nodes {total-number-of-nodes} --nodes-min {minimum-number-of-nodes-in-availability-zone} --nodes-max {maximum-number-of-nodes-availability-zone} --node-volume-size 50 --cluster {cluster-name} --node-private-networking --node-labels=zookeeper=deploy,broker=deploy,grafana=deploy,elasticsearch=deploy,object_store=deploy,distributed_db=deploy,default_agent=deploy,securestore=deploy,monitoring_server=deploy,infra-tsdb=deploy,thanos=deploy,agent1=deploy,northbound=deploy --node-zones {availability-zone-name} --node-type r6i.xlarge --managed

- For resilient setup labels for nodes will be assigned during creation of the nodegroup itself.
- We recommend a minimum of 9 worker nodes; 3 nodes per nodegroup, 1 nodegroup per AZ, and 3 AZs.

Example:

```
eksctl create nodegroup --name aws-deploy-test-ng01 --nodes 3 --nodes-min 3
--nodes-max 3 --node-volume-size 50 --cluster aws-deploy-test
--node-private-networking
--node-labels=zookeeper=deploy,broker=deploy,grafana=deploy,elasticsearch=deploy,obje
ct_store=deploy,distributed_db=deploy,default_agent=deploy,securestore=deploy,monitor
ing_server=deploy,infra-tsdb=deploy,thanos=deploy,agent1=deploy,northbound=deploy
--node-type r6i.xlarge --node-zones us-west-2b --managed

eksctl create nodegroup --name aws-deploy-test-ng02 --nodes 3 --nodes-min 3
--nodes-max 3 --node-volume-size 50 --cluster aws-deploy-test
--node-private-networking
--node-labels=zookeeper=deploy,broker=deploy,grafana=deploy,elasticsearch=deploy,obje
ct_store=deploy,distributed_db=deploy,default_agent=deploy,securestore=deploy,monitor
ing_server=deploy,infra-tsdb=deploy,thanos=deploy,agent1=deploy,northbound=deploy
--node-type r6i.xlarge --node-zones us-west-2c --managed

eksctl create nodegroup --name aws-deploy-test-ng03 --nodes 3 --nodes-min 3
--nodes-max 3 --node-volume-size 50 --cluster aws-deploy-test
--node-private-networking
--node-labels=zookeeper=deploy,broker=deploy,grafana=deploy,elasticsearch=deploy,obje
ct_store=deploy,distributed_db=deploy,default_agent=deploy,securestore=deploy,monitor
ing_server=deploy,infra-tsdb=deploy,thanos=deploy,agent1=deploy,northbound=deploy
--node-type r6i.xlarge --node-zones us-west-2d --managed
```

NOTE: Kindly wait until the required number of nodes are in ready state. You can check the node status by running "*kubectl get nodes*"

- ❏ Create the IAM policy for gp3 volumes creation. Please refer to 5(c) in Prerequisites for respective steps.
- ❏ Update Kubeconfig
  aws eks --region ***<us-west-2>*** *update-kubeconfig --name {cluster-name}*
  Cross check once by running "kubectl get svc --all-namespaces" and observe if kubernetes services are running.

Deploying ATOM

After ensuring that the prerequisites described in the section "Prerequisites" are taken care of, perform the following steps:

1. Login to your Ubuntu machine and connect to cluster and add node labels
   a. Get existing cluster details and connect to it
      "eksctl get cluster"
      "aws eks update-kubeconfig --name {cluster-name} --region {region-name}"

2. Unzip the deployment-scripts folder, provided by Anuta, described in the section, "Deployment scripts and files".
3. Update wrapper.properties file accordingly in [deployment] section.
   a. Cross verify if build number is set
   b. Cross verify if deployment_type is set to **"aws"**
   c. Update size to required value like "minimal" or "resilient"
   d. Cross verify if image_pull is set to "quay"
   e. Cross verify if organization is set to "release"
   f. Set the cluster_name in the [aws-eks] section.
4. Update wrapper.properties file to set ELB configuration in [aws-eks] section
   a. Choose ELB type - if public access to ATOM services is required, then set the following parameters "external". "internal" is default.
      i. elb_atom_ui_sso - ELB for providing ATOM UI over SSO
      ii. elb_atom_ui_direct - ELB for providing ATOM UI/rest API
      iii. elb_atom_agent - ELB for remote agents to connect to ATOM cluster
      iv. elb_atom_agent_apm - ELB for remote agents to send performance metrics to ATOM cluster
      v. elb_atom_agent_incluster - ELB for devices to communicate with agent if the agent is running inside ATOM cluster
   b. set north_bound_source_ranges according to your requirement. By default it will allow all IPs.
      eg: allowed_ip_range_atom_ui = "10.10.0.0/24,172.16.0.0/16"

    i. allowed_ip_range_atom_ui_sso
    ii. allowed_ip_range_atom_ui_direct
    iii. allowed_ip_atom_agent
    iv. allowed_ip_range_atom_agent_apm
    v. allowed_ip_range_atom_agent_incluster

5. Set the permissions to the log file by running. “*sudo touch /var/log/atom.log && sudo chmod 777 /var/log/atom.log*”
6. Modify accesskey,secretkey and service endpoint in ATOM/minio/values.yaml under s3gateway section.

```
s3gateway:
  enabled: s3gateway
  replicas: 1
  serviceEndpoint: "https://s3.{region}.amazonaws.com"
  accessKey: "XXXXXXXXXX"
  secretKey: "YYYYYYYYYYY"
```

7. Deploy ATOM by executing the following script

```
# sudo python deploy_atom.py
```

Executing above steps will complete the ATOM deployment in AWS.
After deployment is completed, the URL’s to access the ATOM application can be fetched by running

```
# kubectl get svc -n atom
```

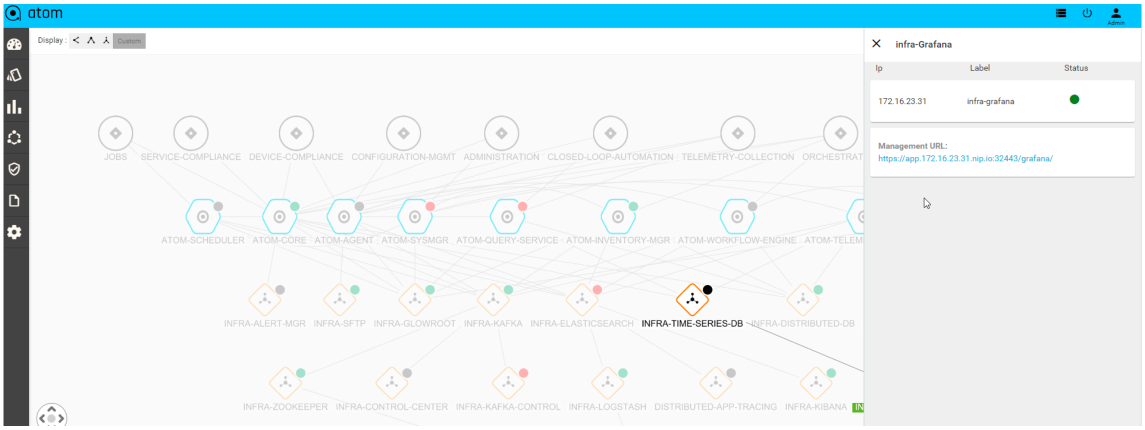
ATOM services are accessible via LoadBalancers.

A Site-to-Site VPN would be needed between the ATOM server on AWS and a remote ATOM agent which has access to lab devices. Please refer to the section Distributed ATOM Agent Deployment for deploying a Remote ATOM Agent.

ATOM System Manager

ATOM provides deployment summary of all Components through System Manager. System Manager provides a high level view of all the components, number of instances, status & management URLs for some of the components.

- Navigate to Administration> System Manager> Dashboard
- To access components like Grafana, Kibana etc. select that component and you can see the Management URL at top right corner.

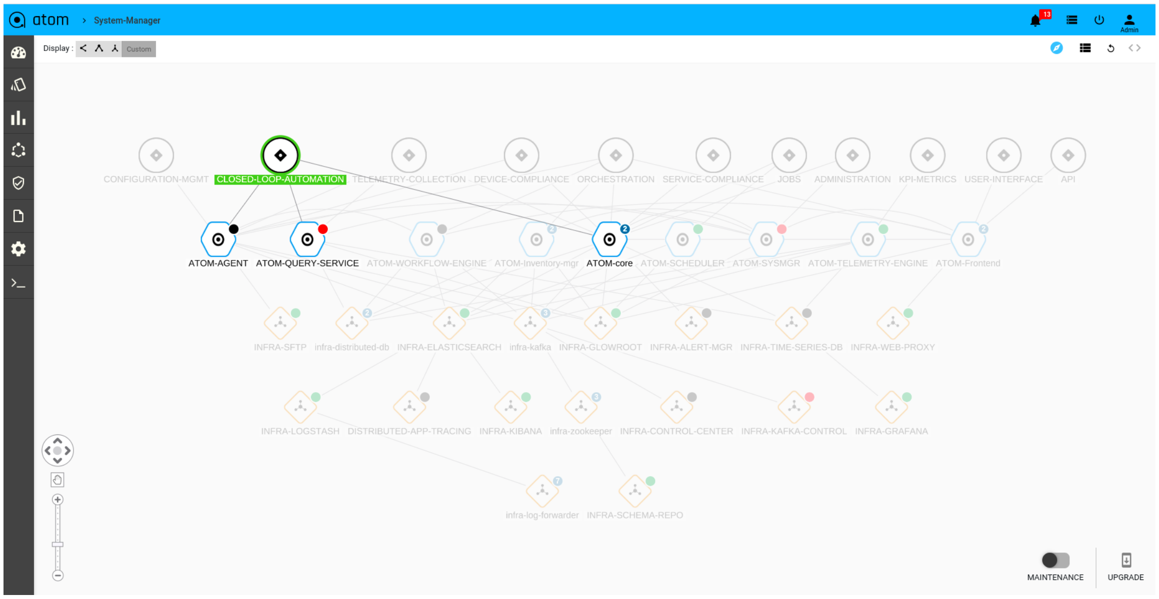


infra-Grafana

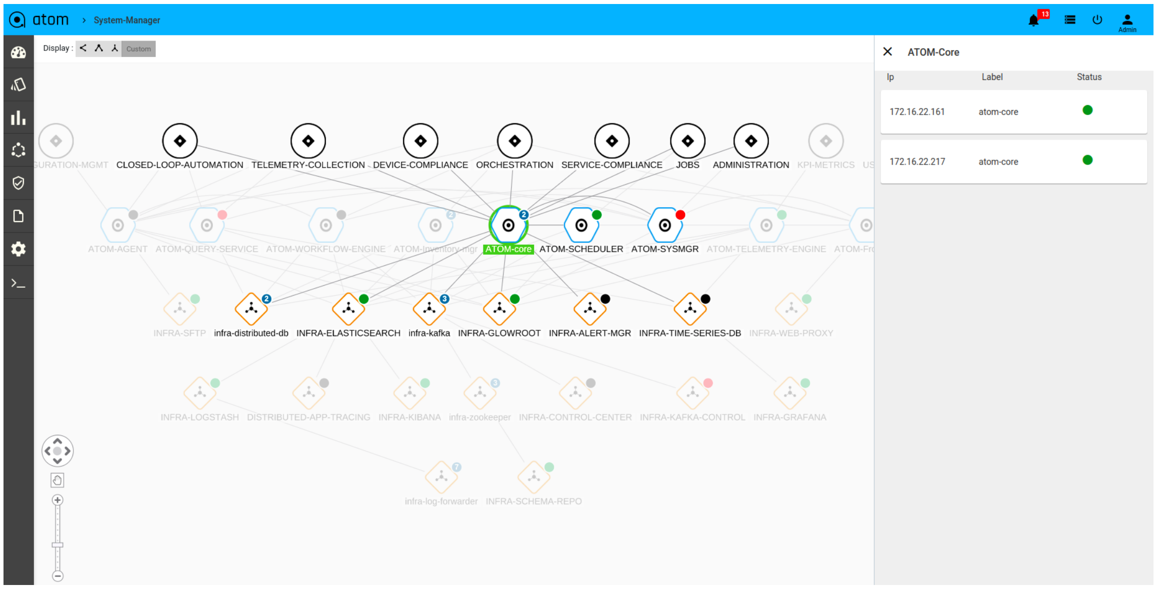
Ip	Label	Status
172.16.23.31	infra-grafana	●

Management URL:
https://app.172.16.23.31.nip.io:32443/grafana/

- To Check the functionality of ATOM and its involved components, select the top level circle icon



- To Check the ATOM Components status like liveness and number of instances per each ATOM components select the ATOM component as shown below.



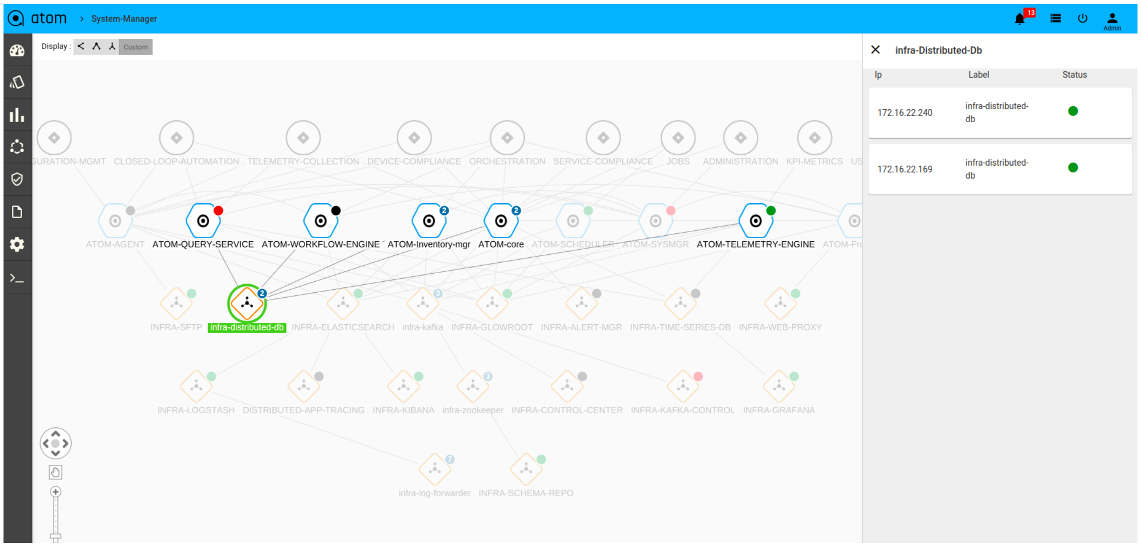
ATOM-Core

Ip	Label	Status
172.16.22.161	atom-core	●
172.16.22.217	atom-core	●

Here on the right side we can see the number instances, refer below colour codings.

1. **Green**: it represent the Activeness of the component
2. **Red**: It represents the component is deployed and the required number of instances set as zero.
3. **Black**: It represents the component is not deployed

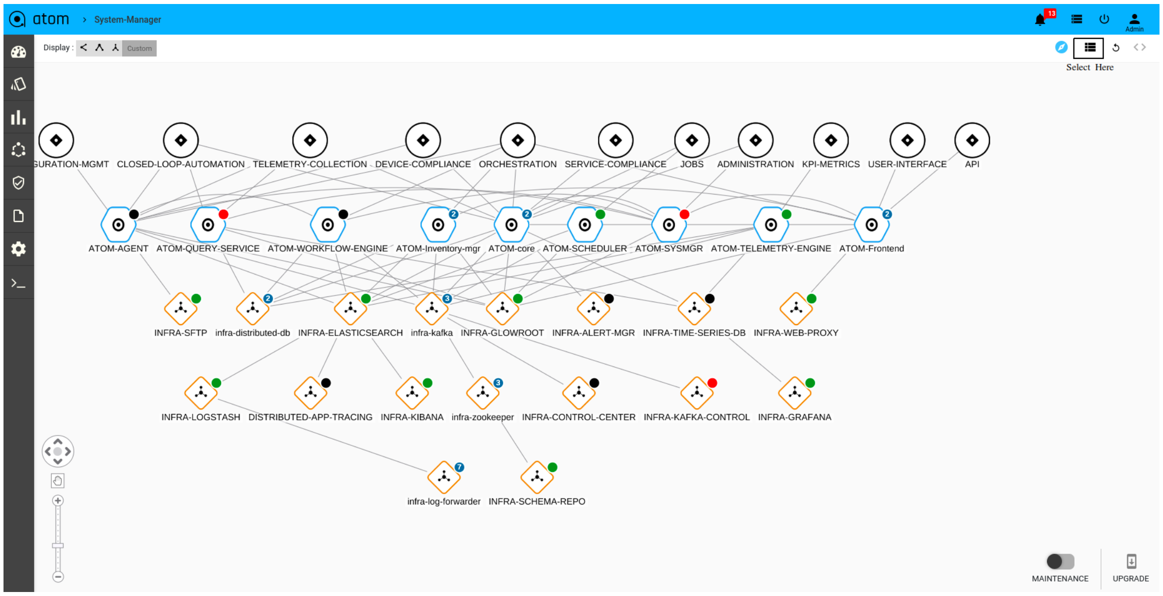
- To Check the ATOM Infra Components status like liveness and number of instances per each Infra components select the Infra component as shown below

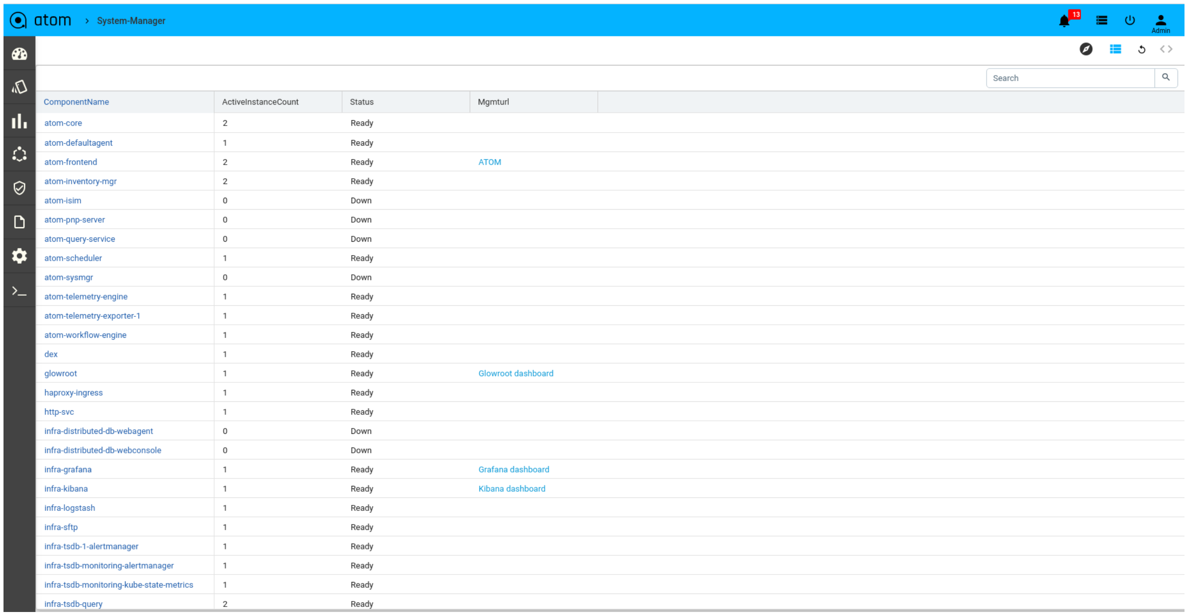


infra-Distributed-Db

Ip	Label	Status
172.16.22.240	infra-distributed-db	●
172.16.22.169	infra-distributed-db	●

- To view the same data in a tabular form select the toggle button as shown below. Here also we can see the number instances per component, status and its management url if applicable.





ComponentName	ActiveInstanceCount	Status	Mgmturl
atom-core	2	Ready	
atom-defaultagent	1	Ready	
atom-frontend	2	Ready	ATOM
atom-inventory-mgr	2	Ready	
atom-isim	0	Down	
atom-pnp-server	0	Down	
atom-query-service	0	Down	
atom-scheduler	1	Ready	
atom-sysmgr	0	Down	
atom-telemetry-engine	1	Ready	
atom-telemetry-exporter-1	1	Ready	
atom-workflow-engine	1	Ready	
dex	1	Ready	
glowroot	1	Ready	Glowroot dashboard
haproxy-ingress	1	Ready	
http-svc	1	Ready	
infra-distributed-db-webagent	0	Down	
infra-distributed-db-webconsole	0	Down	
infra-grafana	1	Ready	Grafana dashboard
infra-kibana	1	Ready	Kibana dashboard
infra-logstash	1	Ready	
infra-sftp	1	Ready	
infra-tsdb-1-alertmanager	1	Ready	
infra-tsdb-monitoring-alertmanager	1	Ready	
infra-tsdb-monitoring-kube-state-metrics	1	Ready	
infra-tsdb-query	2	Ready	

Post Installation

AWS Specific

Security Group updates

Following ports are required to be included in the security groups for device monitoring.
Select VPC under services and select Security Group under Security Section.
VPC – > Security ⟶ Security Group.

select a security group with the name **eks-cluster-sg-{cluster-name}-xxxx** and add following ports to that security group.

- 21(TCP) – FTP from device to ATOM.
- 69(UDP) – TFTP from device to ATOM
- 162(UDP) – Sending SNMP traps from device to ATOM.
- 514(UDP) – Syslog port on device.

the source of the above security group will depend on the device range.

Custom DNS name(CNAME) creation

Once ATOM gets deployed, UI can be accessed at the default DNS names given to the load balancers but those DNS names are not easy to remember and use. If you prefer to use a custom DNS name instead of the default DNS name, you can associate a custom DNS name(CNAME) for the load balancers.

Follow the below steps only if your domain is hosted in aws route53, otherwise follow the instructions from your dns provider to add CNAME.
Steps to add CNAME in route53:

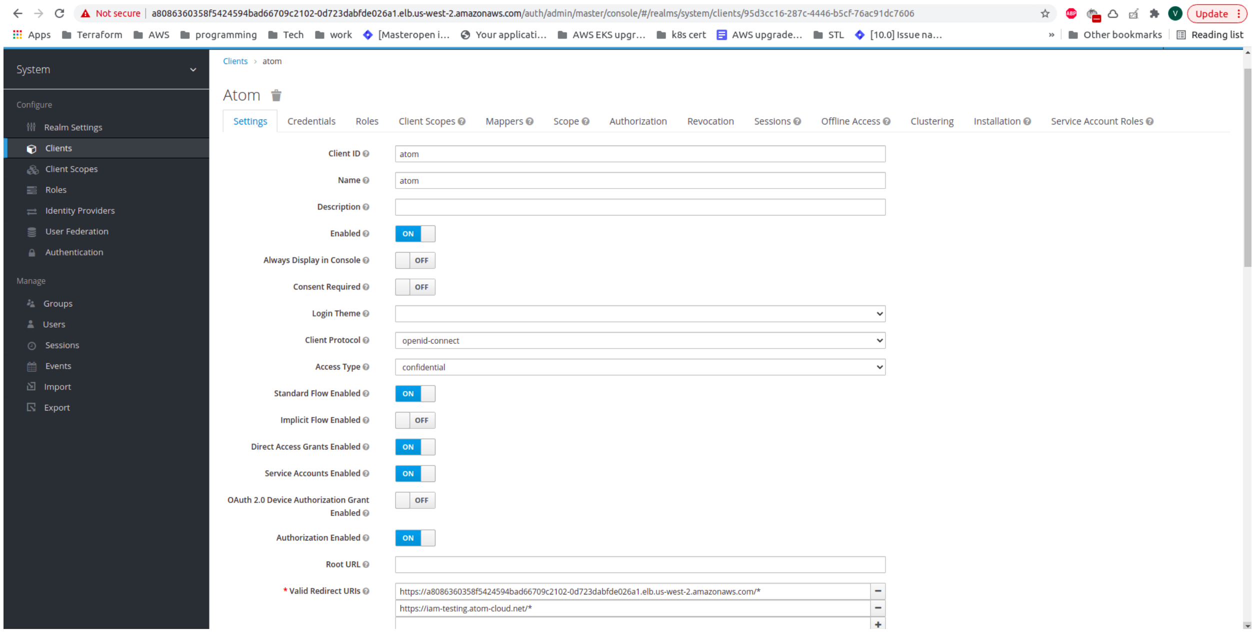
1. Fetch the haproxy-gw load balancer.
    a. kubectl get svc -n atom | grep -i Load | grep haproxy-service-gw
2. Once haproxy LB value fetched, Select **Route53** service in aws and select **HostedZone**
3. In Hosted Zones select one of the domain in which you want to create Record(CNAME type)
4. Click on **Create Record**
    a. In **Record name** provide the name from which you want to access ATOM UI.
    b. Select **CNAME** in Record type.
    c. In the value field , provide the haproxy-service-gw LB value.
    d. Click on Create records.

Once CNAME was added, to make it work changes are required in keycloak UI, atom-frontend and oauth2-proxy deployment.

Adding CNAME in keycloak Clients

Once CNAME was added in Route53, keycloak needs to be updated otherwise it would get InvalidURL when trying to access ATOM UI using CNAME.
steps to follow:

1. Fetch haproxy-service-gw LB value using following command
    a. kubectl get svc -n atom | grep -i Load | grep haproxy-service-gw
2. Access https://{LB}/auth in browser. and click on Administration Console.
3. Select the clients tab under the configure section and click on atom client ID.
4. In Settings, add CNAME value in Valid Redirect URLs and save it.



a8086360358f5424594bad66709c2102-0d723dabfde026a1.elb.us-west-2.amazonaws.com/auth/admin/master/console/#/realms/system/clients/95d3cc16-287c-4446-b5cf-76ac91dc7606

Client ID: atom

Name: atom

Description:

Enabled: ON

Always Display in Console: OFF

Consent Required: OFF

Login Theme:

Client Protocol: openid-connect

Access Type: confidential

Standard Flow Enabled: ON

Implicit Flow Enabled: OFF

Direct Access Grants Enabled: ON

Service Accounts Enabled: ON

OAuth 2.0 Device Authorization Grant Enabled: OFF

Authorization Enabled: ON

Root URL:

* Valid Redirect URIs: https://a8086360358f5424594bad66709c2102-0d723dabfde026a1.elb.us-west-2.amazonaws.com/*
https://iam-testing.atom-cloud.net/*

Deployment changes after adding CNAME

changes are required in atom-frontend and oauth2 deployment to make CNAME work.

1. Modify environment variables in oauth2-proxy deployment
   Steps:
   a. kubectl edit deployment oauth2-proxy -n atom
   b. Replace LB value with CNAME values in following environment variables, OAUTH2_PROXY_OIDC_ISSUER_URL and OAUTH2_PROXY_EXTRA_JWT_ISSUERS

```
Before CNAME:
env:
   -name: OAUTH2_PROXY_OIDC_ISSUER_URL
    value:
https://a8086360358f5424594bad66709c2102-0d723dabfde026a1.elb.us-west
-2.amazonaws.com/auth/key/realms

After CNAME:
env:
    -     name: OAUTH2_PROXY_OIDC_ISSUER_URL
          value: https://iam-testing-atom.cloud.net/auth/key/realms
```

2. Modify environment variables in atom-frontend deployment
   Steps:
   a. kubectl edit deployment atom-frontend -n atom
   b. Replace LB value with CNAME value in env variable KEYCLOAK_URL

Once deployment is successful, you can access the individual microservices running on different nodes of the Kubernetes cluster. Microservices can be accessed via System Manager Dashboard or using access details either via SSO or node-port. For detailed SSO information refer to section ATOM Single Sign-On

Name of the service	Description	How to access it?
Kubernetes Dashboard	Access for Kubernetes dashboard using SSO. Not available for AWS or GKE	https://<*master-ip*>/k8s/
ATOM	Local deployment: Login URL of ATOM UI & SSO based is via master IP Cloud deployment: Kgin URL of ATOM UI via LB of infra-web-proxy service & SSO is via LB of oauth2-proxy service	https://<*master-ip*>:30443 (for local users) https://<*master-ip*>/ (for SSO users) https://<FQDN> (for cloud users)
Grafana	Service which helps us in monitoring infrastructure health using heapster and time series database.	https://<*master-ip*>/grafana/ (for SSO users) https://<FQDN>/grafana (for cloud users)
Kibana	Service which helps us in log monitoring and analytics	https://<*master-ip*>/kibana/ (for SSO users) https://<FQDN>/kibana (for cloud users)

where master_ip is the IP address of the master node and its VIP ip in the case of HA masters setup.

To observe the IP addresses assigned to any of the microservices that could be deployed either on the master or the worker nodes, executing the following commands can help:

```
kubectl get nodes
# Execute this command to view the created nodes

kubectl get pods -n atom
# Execute this command to get the microservices deployed on the node

kubectl describe pod <pod_name> -n atom
#Execute this command to view the details of the microservice

kubectl get svc -n atom
#Execute this command to view the services
```

NOTE: If deployment is done in a different namespace, provide `-n <namespace>` in the above commands.

ATOM Single Sign-On (SSO)

ATOM Single Sign-On supports following Identity Providers (IdPs).

- Keycloak - Keycloak is an open source identity provider and runs within the ATOM cluster.

For Log in with Atom, default user/password: admin/Secret@123 can be used. Additional users can be created by login to the atom authentication manager ui https://<master-ip>/auth.

- Google - Please go to section Google Idp of this guide to configure the integration with Google SSO.

ATOM SSO support is enabled by default with the Keycloak as Identity Provider running locally in the ATOM cluster. Additional steps are needed to configure integration with different IdPs such as Google.

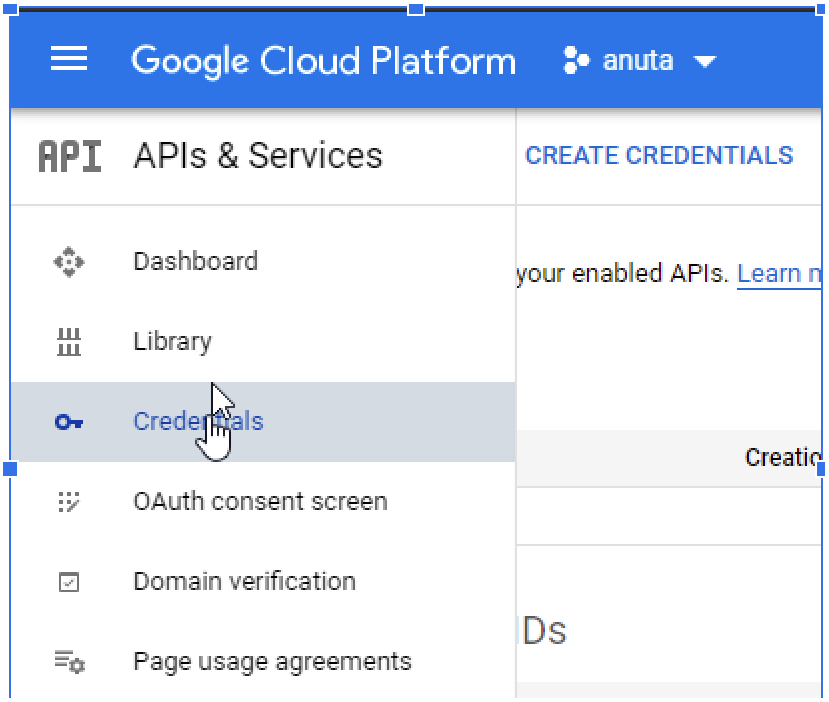
Default *<master_ip>* is configured during ATOM setup with self signed certificates. If a specific domain is desired, then users can provide a FQDN address for the master IP of K8s cluster and an SSL certificate associated with the same FQDN in PEM format.

Below are the steps listed for setting up **Google SSO** Integration with ATOM.

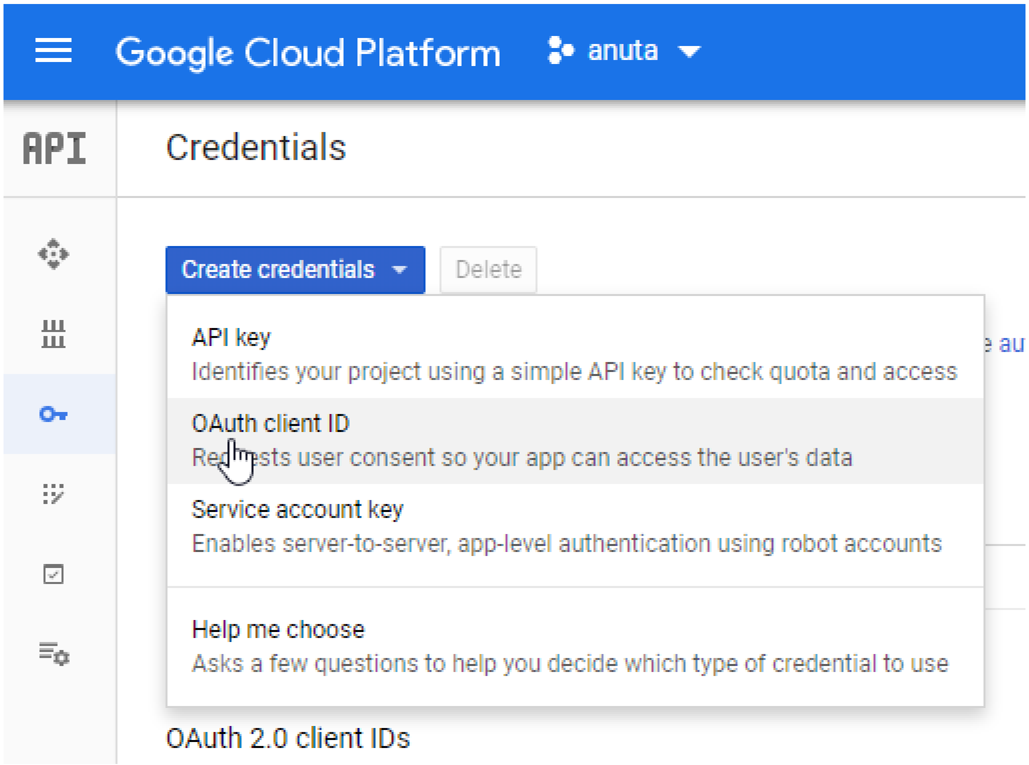
Google IdP

1. For Google based SSO logins, create callback urls in Google cloud platform.

   - Login to Google cloud platform. In the project Dashboard center pane, choose APIs & Services tab.
   - In the left navigation pane, choose "Credentials"



   - Click the create credentials button. Select OAuth client ID.



API key
Identifies your project using a simple API key to check quota and access

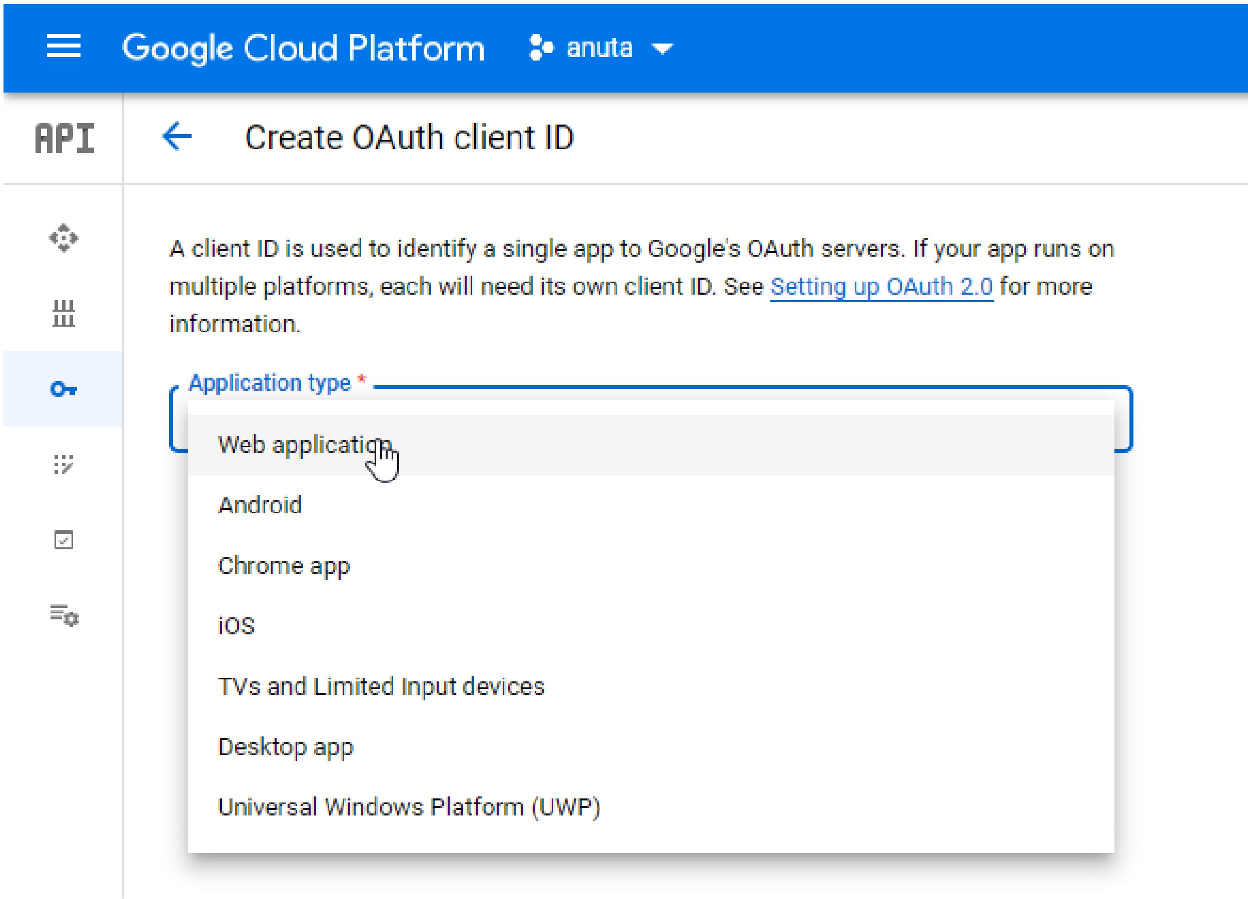
OAuth client ID
Requests user consent so your app can access the user's data

Service account key
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

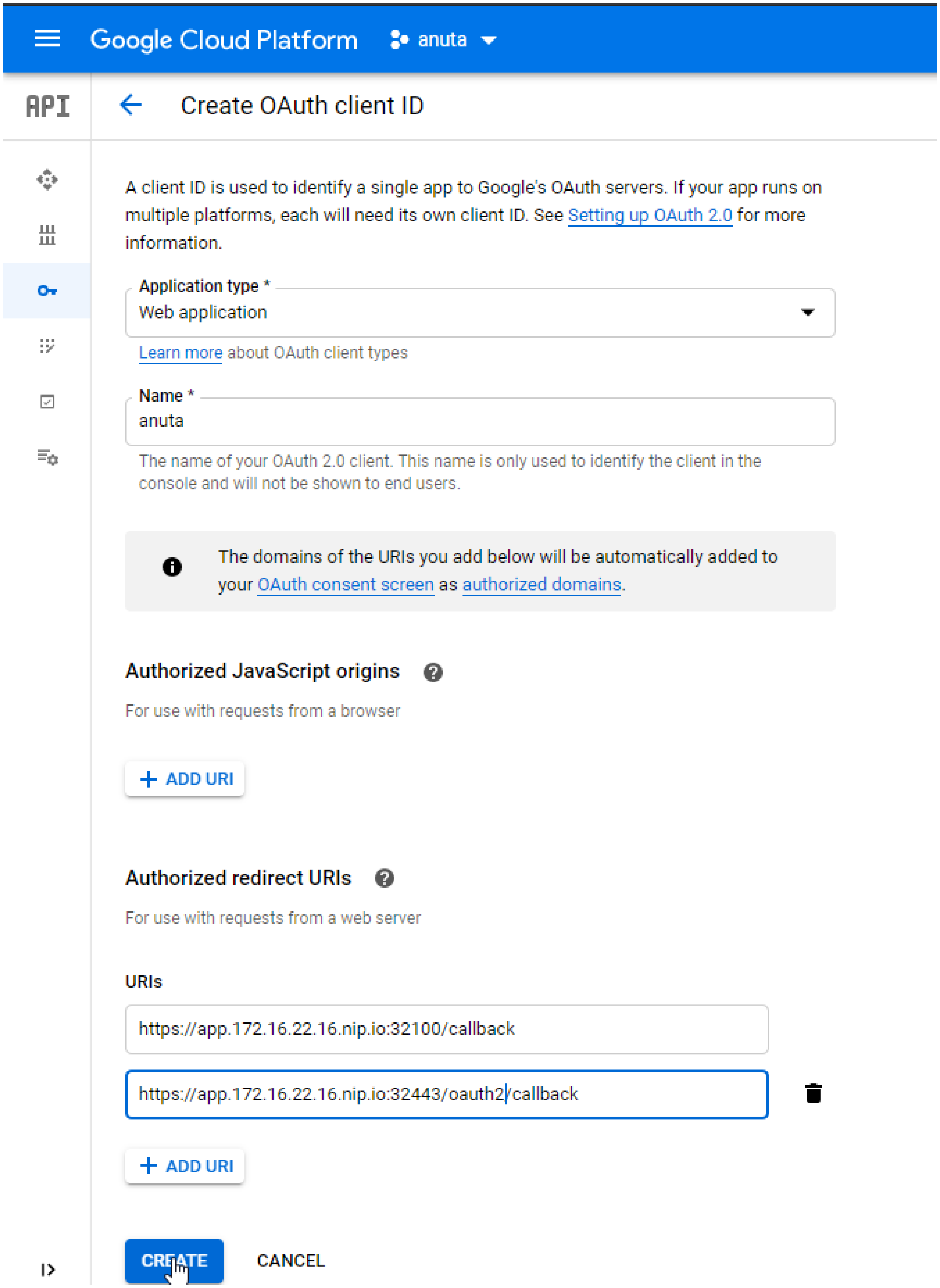
OAuth 2.0 client IDs

- Select application type as web application



A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information.

- Give name, add callback urls like below under Authorized redirect Urls and click on the create button.after that copy client id and secrets.



Create OAuth client ID

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information.

Application type *
Web application

Learn more about OAuth client types

Name *
anuta

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins

For use with requests from a browser

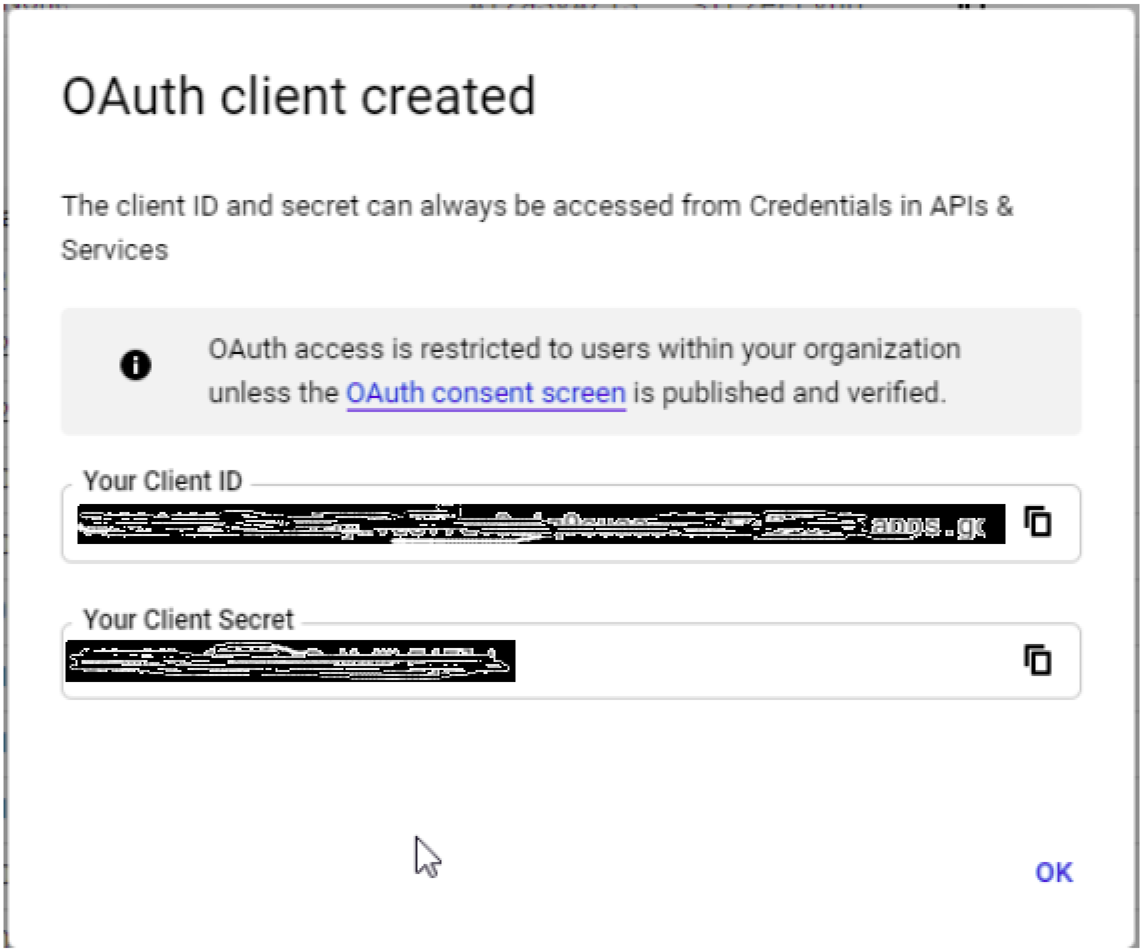
Authorized redirect URIs

For use with requests from a web server

URIs

https://app.172.16.22.16.nip.io:32100/callback

https://app.172.16.22.16.nip.io:32443/oauth2/callback



OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

OAuth access is restricted to users within your organization unless the OAuth consent screen is published and verified.

Your Client ID

[redacted]

Your Client Secret

[redacted]

- Login to keycloak and update Identity provider details client id and client secret
- Now we can login to the atom application using google credentials.

ATOM System Alerts

Below are a list of ATOM System Alerts and scenarios when they can be generated, Actions which can be taken.

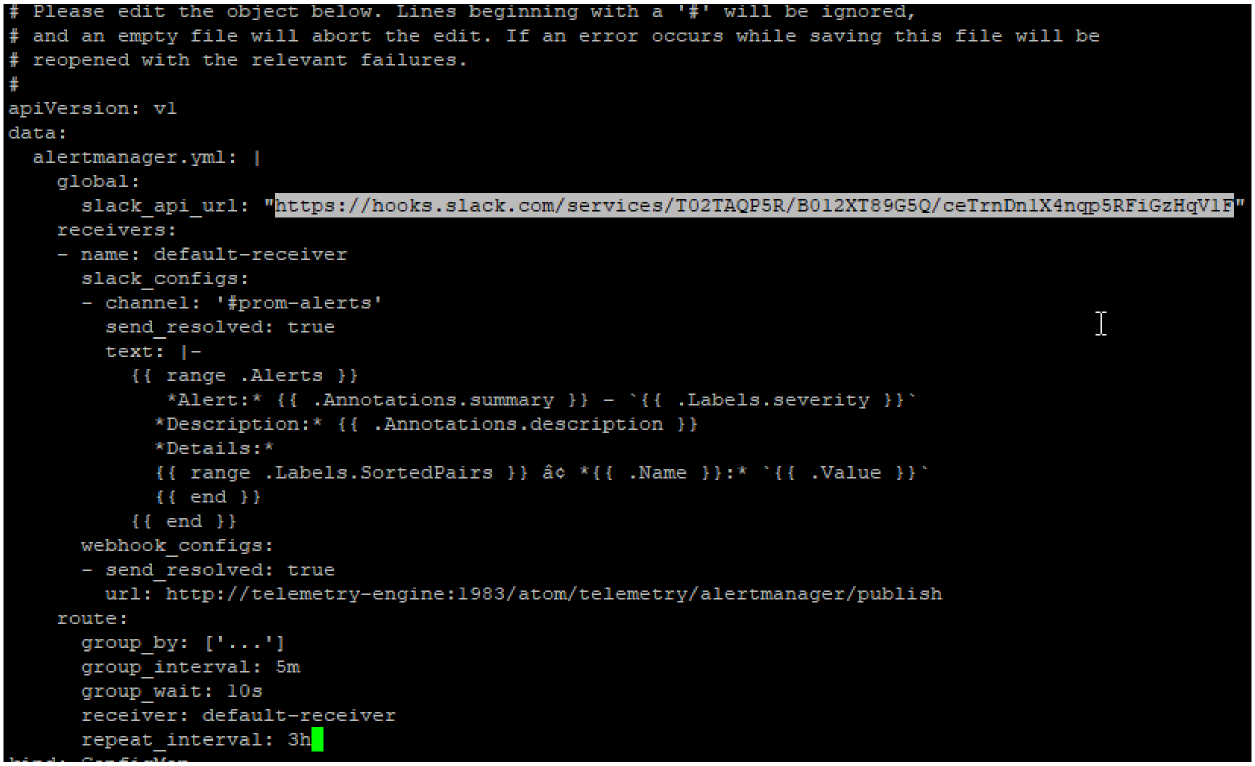
System Alert Name	Troubleshooting Steps
NodeHighMemory	● Login to Grafana (https://<Master IP>/grafana/) ● Select Cluster Health dashboard ● Select the node which has a HighMemory alert and check which are the pods consuming more memory in that node.
NodeHighCPU	● Login to Grafana (https://.<Master IP>/grafana/) ● Select Cluster Health dashboard ● Select the node which has HighCPU alert and check which are the pods consuming more CPU.

NodeHighDiskUsage	● Login to Grafana. ● Select Cluster Monitoring Dashboard. ● Select the node which has High Disk usage and login to that node. ○ Login to Master VM ○ Find the IP of the node by following command. ■ **kubectl describe node <node-name> | grep IP**. you will get the IP of that node and then you can login to that node by using ssh. ● Check which folders are consuming Disk in /data folder by using **du -h --max-depth=1** on that particular node. ● Run purge jobs in atom to cleanup the disk. ● If /data folder is consuming less then what shown in grafana then check the disk usage by following command ○ **df -h** which will give full disk utilization. (/dev/mapper/centos-root is the filesystem we need to check)
NodeFault	This alert can be received for the following reasons. ● MemoryPressure: if pressure exists on the node memory (if the node memory is low) ● PIDPressure: if pressure exists on the processes (if there are too many processes on the node) ● DiskPressure: if pressure exists on the disk size (if the disk capacity is low) ● NetworkUnavailable: if the network for the node is not correctly configured. If you receive this Alert follow the steps below. ● Note the fault condition and node ● To get the list and status of nodes - **kubectl get nodes**. ● To check the fault condition type - **kubectl describe <node-name>**. In the output you will see the reason for the failure in conditions. Can be due to DiskPressure/Memory etc.
InstanceDown	This alert will come when the Node/VM is not reachable or down. If you receive this Alert, intimate to Admin.
PodNotReady	Pod can be in NotReady for a number of reasons. To get an overview of all the pods in ATOM execute **kubectl get pods -n atom**. To find out the reason why the specific pod was not ready execute **kubectl describe pod <pod-name> -n atom** Following can be the reasons for PodNotReady ● Taints on Node: If taints were added on node and pod spec doesn't have tolerations w.r.t to taints on node then Pod will be in pending state.

	● Insufficient Resources: If there are resource crunch in the cluster or required resources to deploy the pod was not available on the cluster then the pod will be in pending state. ● Node Selector: If the pod spec has nodeselector then for the pod to be eligible to run on a node, then node must have each of the indicated key-value pairs which is mentioned in nodeselector as labels. ● PV Claim: If scheduler doesn't find node labels to deploy PV then pod will be in pending status with error "pod has unbound immediate PersistentVolumeClaims". ● InitStuck: If a pod is stuck at init phase then subsequent init-containers are not ready. This could be because of the dependent pod being down. Ex: schema-repo dependent on broker.
ContainerNotUp	Below can be possible reasons for Container not being up. ● Crashloopbackoff ● ImagePullbackoff ● Application inside the container was not up. When this Alert is generated follow the steps below. ● Note the reason shown for the Container not up and pod-name ● Login to the Kibana (https://<Master IP>/kibana/) ● Filter by pod name ● Verify the logs and find if any error messages are shown.
ContainerTerminated	Containers can be killed for a number of reasons like OutofMemory (OOMKilled), Eviction, DiskPressure etc.. When this Alert is generated follow the steps below. ● Note the reason shown for the Container termination and pod-name ● Login to the Kibana (https://<Master IP>/kibana/) ● Examine kubernetes logs - use the query 'pod-name : "eventrouter' and select the appropriate time range, look for related logs. Additional filtering criteria like name of the pod can be used in the query. ● Look for errors or warnings in Pod logs - Filter by pod name. Ex- '"pod-name: "atom-core" and (error or warning or warn)'
ReplicasMismatch	This alert can be received when one or more pods are ● Not ready due to crashed application internally ● In pending state due to missing resource ● In failed container state When this Alert is generated follow the steps below ● Check the cause using **kubectl logs <pod> -n atom** and **kubectl describe <pod> -n atom**

ATOM System Alerts by default can be observed in ATOM > Assurance > Alerts. To enable those alerts to come into Slack as well, make sure to update the webhook url of the slack channel in config map of infra-tsdb-monitoring-alertmanager by following below steps:

- Login to the Master IP through ssh and execute below command.
  kubectl edit configmap infra-tsdb-monitoring-alertmanager -n atom



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  alertmanager.yml: |
    global:
      slack_api_url: "https://hooks.slack.com/services/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"
    receivers:
    - name: default-receiver
      slack_configs:
      - channel: '#prom-alerts'
        send_resolved: true
        text: |-
          {{ range .Alerts }}
             *Alert:* {{ .Annotations.summary }} - `{{ .Labels.severity }}`
            *Description:* {{ .Annotations.description }}
            *Details:*
            {{ range .Labels.SortedPairs }} â¢ *{{ .Name }}:* `{{ .Value }}`
            {{ end }}
          {{ end }}
      webhook_configs:
      - send_resolved: true
        url: http://telemetry-engine:1983/atom/telemetry/alertmanager/publish
    route:
      group_by: ['...']
      group_interval: 5m
      group_wait: 10s
      receiver: default-receiver
      repeat_interval: 3h
```

- Update your slack webhook url in **slack_api_url** option.
- As intention is to have alerts observed in both slack and atom ui, make sure **receiver** field value is *“**default-receiver**”*
- **group_by:** It is useful for alert deduplication and repeatability or stacking the alerts together.
  - [...] treats every label name and value as different, don’t change this unless you want different behavior.
    - For example: if you keep [device] as group_by attribute then each device alert will be notified only one irrespective of its type, severity etc..
- Tune below timers based on your requirement, however default values are sufficient to get all the notifications.
  - **group_wait** : How long to wait to buffer alerts of the same group before sending a notification initially. Usually, it will o to few minutes
  - **group_interval** :How long to wait before sending an alert that has been added to a group for which there has already been a notification. Usually, it is 5 or more minutes
  - **repeat_interval** : How long to wait before re-sending a given alert that has already been sent in a notification. Usually, it depends on the SLA’s to acknowledge and resolve the issues in your environment. Don’t keep less than 1 hour, as it chokes the system with too many duplicate notifications.

  Note: Please do understand each option before changing from default to any other values as it impacts the throttling, alert deduplication.

Troubleshooting & FAQ

Following can be some issues seen during deployment.

Issue	Troubleshooting Steps
ATOM UI page is not reachable	1. Check if all the VM nodes in Esxi are in powered-on state 2. Login to the Master and Check if K8s cluster shows all Nodes are in READY state 3. Login to Master and Check if all the ATOM pods are in Status:Running state
ATOM UI page shows: 503 Service Unavailable	1. Login to Master and Check if all the ATOM pods are in Status:Running state. 2. Check if all the Pods are showing READY 1/1, 2/2, 3/3 as applicable based on containers it holds.
ATOM deployment on KVM, where low CPU and I/O performance can impact	Cross check the CPU pinning if required and set the I/O mode to “native” in the node’s xml file
Overlapping IP address issue during ATOM deployment.	Calico CNI from Anuta defaults to 10.200.0.0/16 for the pods (containers). So one needs to cross check their lab networking before forming a kubernetes cluster.
Accessibility test between Remote Agent and ATOM Server over required nodePorts	To check the accessibility of databases running on ATOM Server from remote agent, one can run curl to one of the endpoints(nodePorts) like “curl -v http://<ATOM node IP>:<nodePort>” E.g: curl -v http://172.16.100.10:30081
Service Loadbalancers are in pending state	Check if metallb pods are running and active. If they are missing, then install the metallb helm chart. Prior to installation, set the VIP’s at values.yaml file

List of useful commands

Some of the commands that will be useful for debugging and troubleshooting.

Command	Description
`helm create {package_name}`	To create a directory
`helm install {package_name} -n {name of app}`	Deploy the application
`helm ls -n atom`	To check deployment status

Kubectl get deployments -n atom	To check deployments
Kubectl get pods -n atom	To check pod status
helm upgrade {releasename} {package-name} -n atom	deployment update with new changes.
helm history {package-name} -n atom	To view the history
helm uninstall {package_name} -n atom	To delete the package
helm rollback {package-name} version -n atom	Rollback the package

Cleanup of Deployment

By keeping the Kubernetes cluster, If ATOM Server deployment needs to be deleted for recreating it, then a proper cleanup needs to be done by following the below steps on the Kubernetes master.

1. Execute below cmd
   helm uninstall `helm ls -n atom | awk 'NR>1 {{print $1}}'` -n atom
2. From scripts folder of atom-deployment zip, execute 'sh teardown-pv-pvc.sh'
3. From scripts folder of atom-deployment zip, execute 'sh script_delete.sh'
4. Check if all the deployments got deleted or not by executing
   - kubectl get deployments -n atom
   - kubectl get statefulsets -n atom
   - kubectl get pods -n atom
   - helm ls -n atom
   - kubectl get pv,pvc -n atom
5. Once all cleanup is done, execute 'sudo python deploy_atom.py' from Master node.

Guidance on KVM

Make sure you have the qcow images from Anuta or convert from OVA like below.

```
tar -xvf centos_1_21_300_linstor_0122.ova
```

# Convert the vmdk to qcow2

```
sudo qemu-img convert -f vmdk -O qcow2 centos_1_21_300_linstor_0122-disk1.vmdk
centos_1_21_300_linstor_0122-disk1.qcow2

sudo qemu-img convert -f vmdk -O qcow2 centos_1_21_300_linstor_0122-disk2.vmdk
centos_1_21_300_linstor_0122-disk2.qcow2
```

If you are working on a remote KVM machine without a GUI tool like vm manager, follow below steps

- Dedicated Master node

```
virt-install --name "<VM_NAME>" --memory <RAM_IN_MB> --vcpus <CPU_COUNT> --disk
<FULL_PATH_OF_MASTER_QCOW2_IMAGE>,bus=virtio --network=<BRIDGE_NAME_AND_TYPE>
--vnc --import --nographics --os-type=linux --os-variant=centos7.0
```

```
ex:
virt-install --name "master" --memory 8192 --vcpus 4 --disk
/home/anuta/Downloads/master/centos_1_21_40.qcow2,bus=virtio
--network=bridge:virbr0,model=virtio --vnc --import --nographics
--os-type=linux --os-variant=centos7.0
```

- For each Shared Master or Worker nodes with linstor disks

```
virt-install --name "<VM_NAME>" --memory <RAM_IN_MB> --vcpus <CPU_COUNT> --disk
<DISK1_QCOW2_IMAGE>,bus=virtio --disk <DISK2_QCOW2_IMAGE>,bus=virtio
--network=<BRIDGE_NAME_AND_TYPE> --vnc --import --nographics --os-type=linux
--os-variant=centos7.0
```

```
ex:
virt-install --name "worker1" --memory 32768 --vcpus 4 --disk
/home/anuta/Downloads/worker1/centos_1_21_300_linstor_1221-disk1.qcow2,bus=virt
io --disk
/home/anuta/Downloads/worker1/centos_1_21_300_linstor_1221-disk2.qcow2,bus=virt
io --network=bridge:virbr0,model=virtio --vnc --import --nographics
--os-type=linux --os-variant=centos7.0
```

In the case of Node having Multiple disks, make sure that `centos_1_21_300_linstor_1221-disk1.qcow2` is used for booting the VM. For this the disks need to be mapped appropriately to the correct name.

Boot disk always maps to `centos_1_21_300_linstor_1221-disk1.qcow2` which maps to vda, while Data disk always maps to `centos_1_21_300_linstor_1221-disk2.qcow2` which maps to vdb. If we still find that the VM does not boot appropriately a quick troubleshooting step would be to try and boot from the other disk.

Migration of Storage

Please follow the following steps to migrate Nodes from HDD to SSD or other suitable storage options.

STEP-1 ATOM prerequisites before Data store migration:

1. Put ATOM In maintenance mode.
   Navigate to Administration > System Manager > Dashboard. Enable “Maintenance” option to put the system in maintenance mode.
2. Shutdown all nodes (VMs) that need to be migrated

STEP-2 ATOM prerequisites before Data store migration:

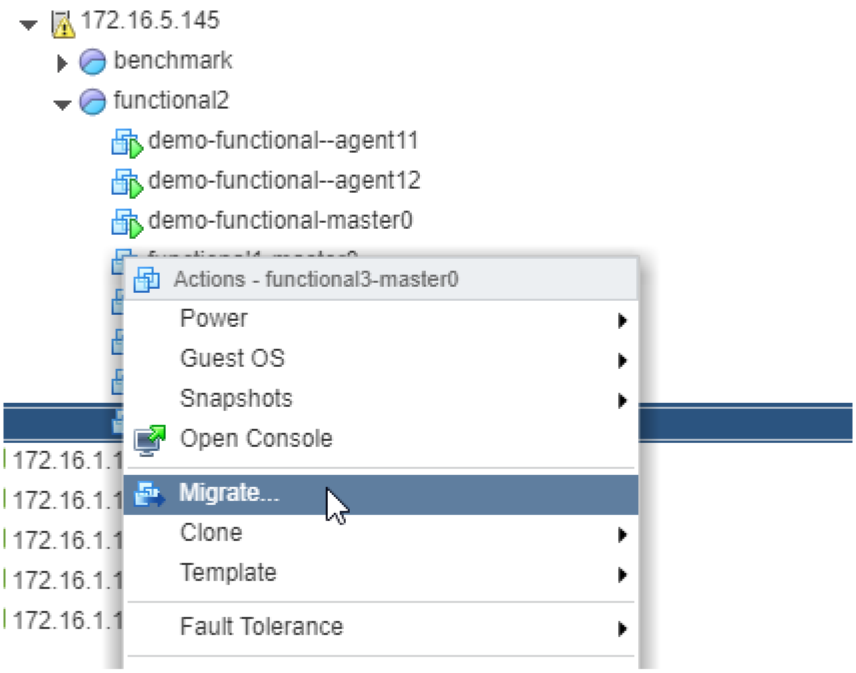
1. Migrate VM and change the Data Storage - for example, nodes running on esxi hosts can be migrated using vSphere.
2. Power on the Nodes

STEP-3 Post VM Migration Steps in ATOM:

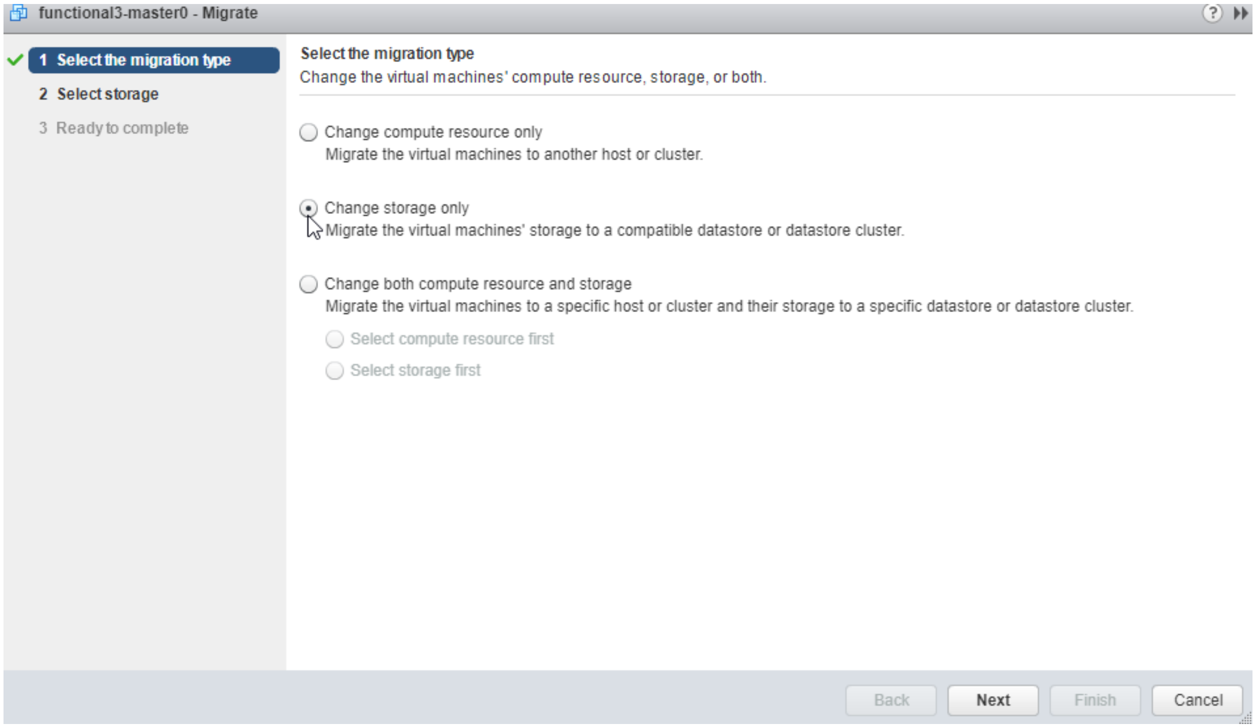
1. Remove the maintenance mode.
   Navigate to Administration > System Manager > Dashboard. Disable "Maintenance" option to clear the system from maintenance mode.

Following Example shows STEP-2 in a VMware based virtualization environment

1. Right click on Node(VM) and select migrate option.



2. Select the migration type.



Select the migration type
Change the virtual machines' compute resource, storage, or both.

○ Change compute resource only
Migrate the virtual machines to another host or cluster.

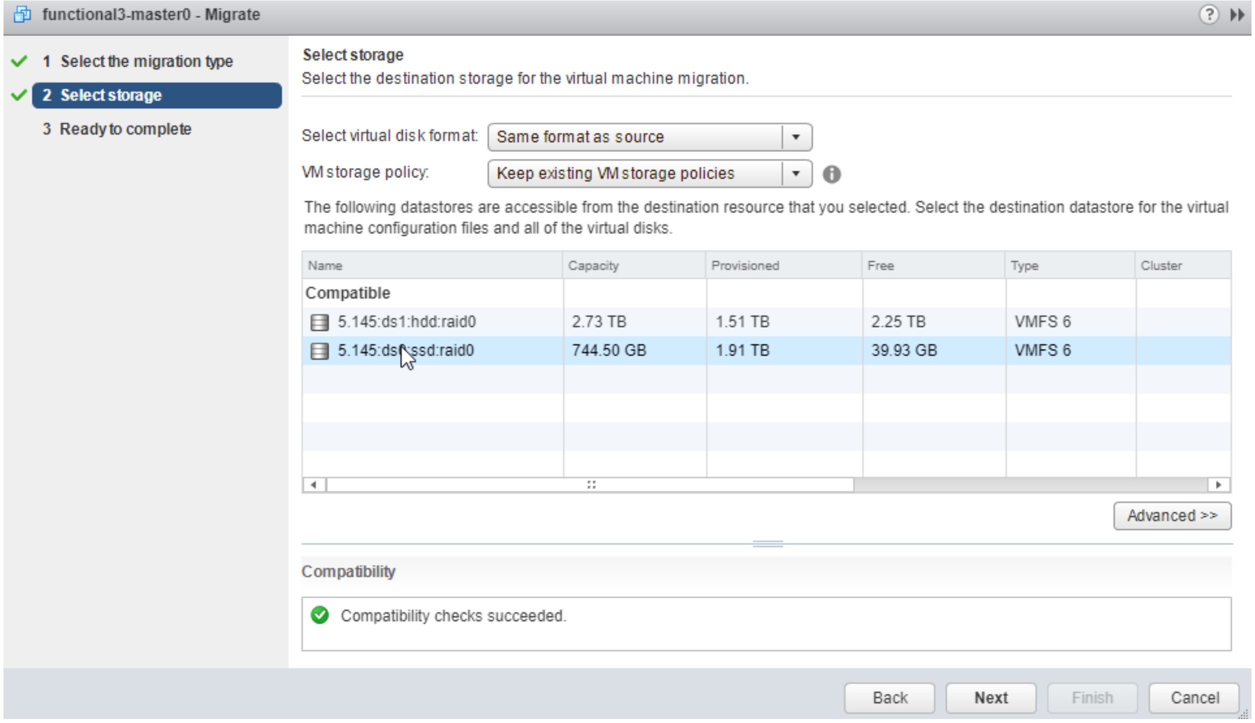
● Change storage only
Migrate the virtual machines' storage to a compatible datastore or datastore cluster.

○ Change both compute resource and storage
Migrate the virtual machines to a specific host or cluster and their storage to a specific datastore or datastore cluster.

○ Select compute resource first

○ Select storage first

3. Select the storage type.



functional3-master0 - Migrate

Select storage
Select the destination storage for the virtual machine migration.

Select virtual disk format: Same format as source

VM storage policy: Keep existing VM storage policies

The following datastores are accessible from the destination resource that you selected. Select the destination datastore for the virtual machine configuration files and all of the virtual disks.

Name	Capacity	Provisioned	Free	Type	Cluster
Compatible					
5.145:ds1:hdd:raid0	2.73 TB	1.51 TB	2.25 TB	VMFS 6	
5.145:ds1:ssd:raid0	744.50 GB	1.91 TB	39.93 GB	VMFS 6	

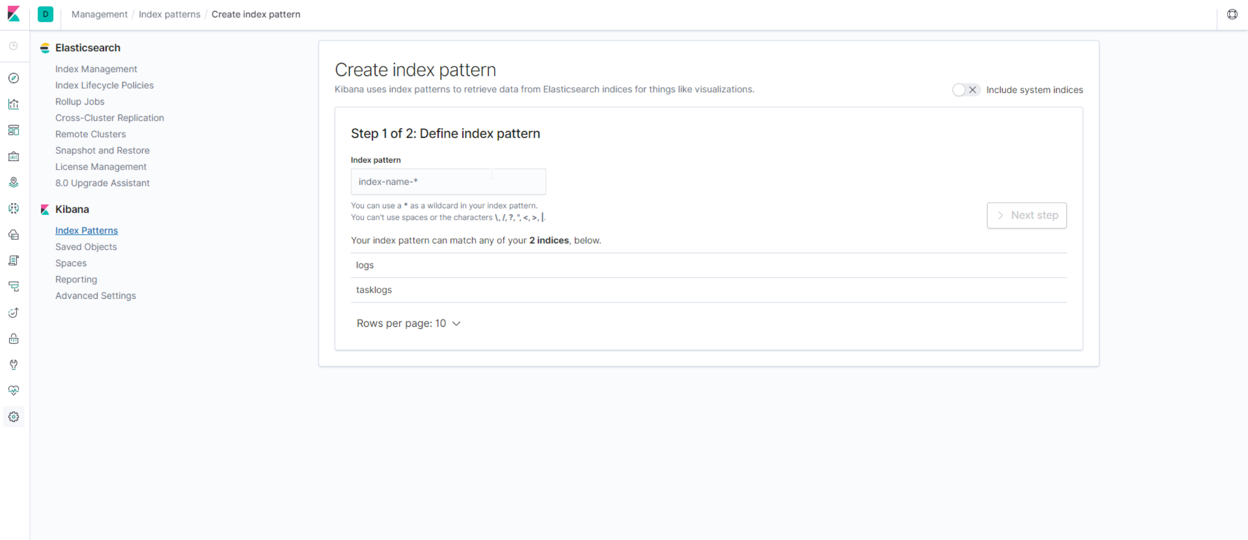
Compatibility

Compatibility checks succeeded.

4. Click on the Next and Finish button.

Steps to check logs in kibana

1. Open the kibana url , http://<master_ip>/kibana/
2. Create index pattern by going to Management/Index patterns/Create index pattern



Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern: index-name-*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

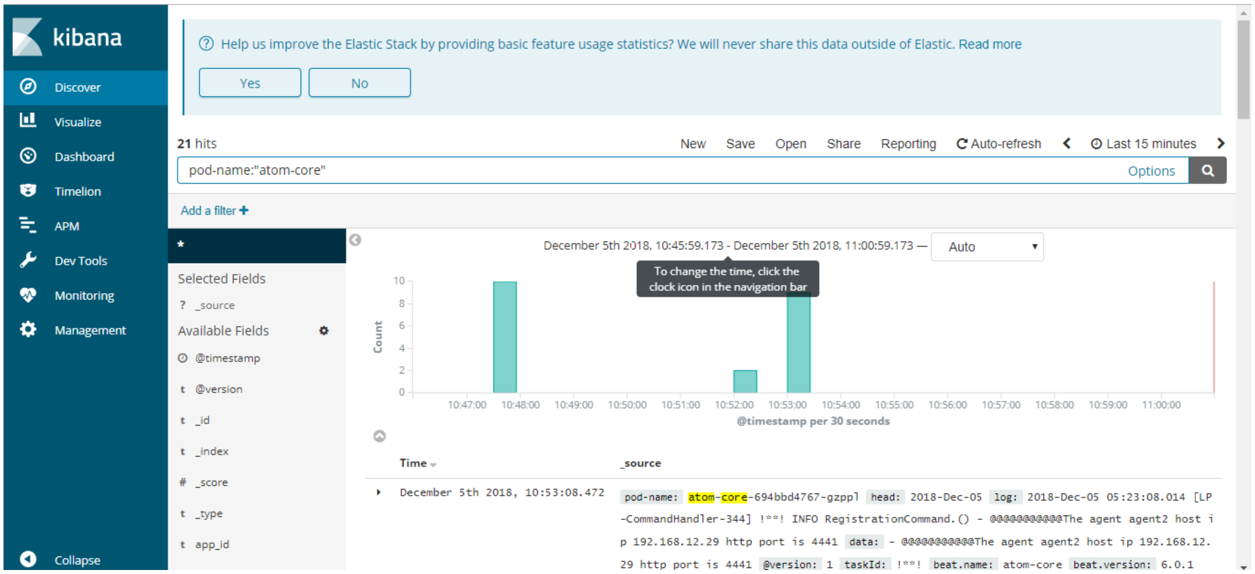
Your index pattern can match any of your **2 indices**, below.

logs

tasklogs

Rows per page: 10

3. Go to Discover and in the search box , search with pod name as shown below pod-name:”<pod-name>”, to check logs for specific pod



Help us improve the Elastic Stack by providing basic feature usage statistics? We will never share this data outside of Elastic. Read more

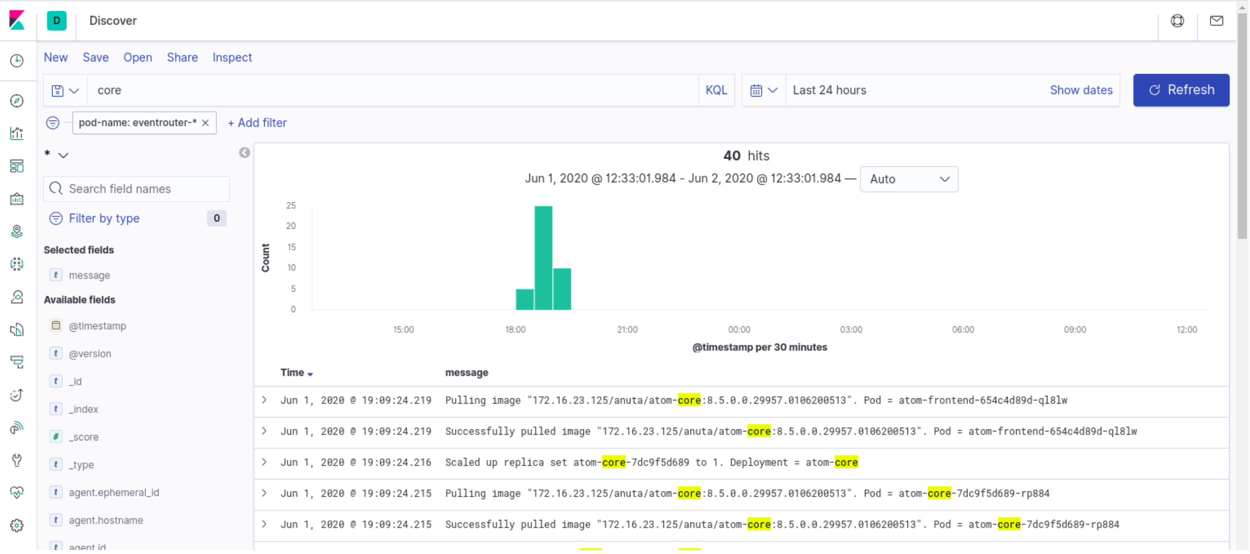
21 hits

pod-name:"atom-core"

December 5th 2018, 10:45:59.173 - December 5th 2018, 11:00:59.173 —

```
Time         _source
December 5th 2018, 10:53:08.472   pod-name: atom-core-694bbd4767-gzppl head: 2018-Dec-05 log: 2018-Dec-05 05:23:08.014 [LP
                                  -CommandHandler-344] !**! INFO RegistrationCommand.() - @@@@@@@@@@@The agent agent2 host i
                                  p 192.168.12.29 http port is 4441 data: - @@@@@@@@@@@The agent agent2 host ip 192.168.12.
                                  29 http port is 4441 @version: 1 taskId: !**! beat.name: atom-core beat.version: 6.0.1
```

4. Some useful queries to get K8s events and K8s logs
   For atom core pod events



core

Last 24 hours

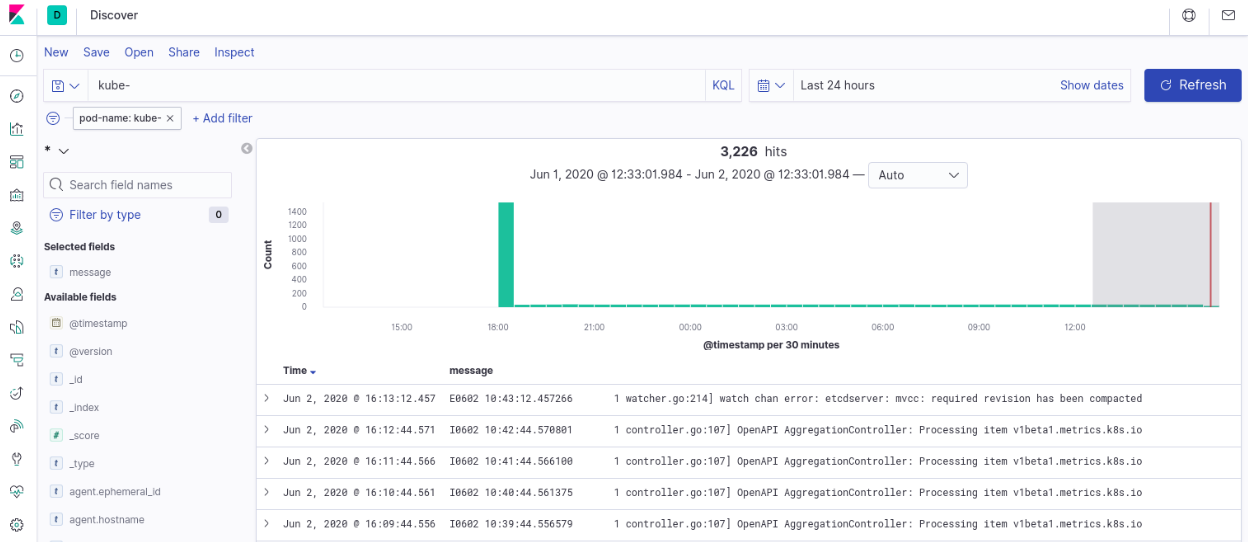
pod-name: eventrouter-*

40 hits

Jun 1, 2020 @ 12:33:01.984 - Jun 2, 2020 @ 12:33:01.984 —

```
Time                        message
Jun 1, 2020 @ 19:09:24.219  Pulling image "172.16.23.125/anuta/atom-core:8.5.0.0.29957.0106200513". Pod = atom-frontend-654c4d89d-ql8lw
Jun 1, 2020 @ 19:09:24.219  Successfully pulled image "172.16.23.125/anuta/atom-core:8.5.0.0.29957.0106200513". Pod = atom-frontend-654c4d89d-ql8lw
Jun 1, 2020 @ 19:09:24.216  Scaled up replica set atom-core-7dc9f5d689 to 1. Deployment = atom-core
Jun 1, 2020 @ 19:09:24.215  Pulling image "172.16.23.125/anuta/atom-core:8.5.0.0.29957.0106200513". Pod = atom-core-7dc9f5d689-rp884
Jun 1, 2020 @ 19:09:24.215  Successfully pulled image "172.16.23.125/anuta/atom-core:8.5.0.0.29957.0106200513". Pod = atom-core-7dc9f5d689-rp884
```

For K8s pod logs



kube-

Last 24 hours

pod-name: kube-

3,226 hits

Jun 1, 2020 @ 12:33:01.984 - Jun 2, 2020 @ 12:33:01.984 —

```
Time                        message
Jun 2, 2020 @ 16:13:12.457  E0602 10:43:12.457266       1 watcher.go:214] watch chan error: etcdserver: mvcc: required revision has been compacted
Jun 2, 2020 @ 16:12:44.571  I0602 10:42:44.570801       1 controller.go:107] OpenAPI AggregationController: Processing item v1beta1.metrics.k8s.io
Jun 2, 2020 @ 16:11:44.566  I0602 10:41:44.566100       1 controller.go:107] OpenAPI AggregationController: Processing item v1beta1.metrics.k8s.io
Jun 2, 2020 @ 16:10:44.561  I0602 10:40:44.561375       1 controller.go:107] OpenAPI AggregationController: Processing item v1beta1.metrics.k8s.io
Jun 2, 2020 @ 16:09:44.556  I0602 10:39:44.556579       1 controller.go:107] OpenAPI AggregationController: Processing item v1beta1.metrics.k8s.io
```

Steps to check load distribution in kafka for config parser

Login to one of the brokers and execute below commands.

```
export KAFKA_JVM_PERFORMANCE_OPTS=""

kafka-consumer-groups --bootstrap-server localhost:9092 --describe
--group config-parser
```

It shows all partition details along with consumer and lag details

Logs for deployment failures

From the master node execute `getlogsfrompod.sh` shell script. The script is available in the scripts folder of atom-deployment zip.

```
# cd scripts
# sh getlogsfrompod.sh
```

- This script creates a .tgz in /tmp folder.
- Also collect /var/log/atom.log from master node.

Appendix

AWS Connectivity Options

Connectivity between DC and AWS:

AWS provides different options to connect the on-premises site to aws. We recommend the customer to follow the existing practice if there is one setup already. Here is the aws reference doc to consider different possible options. https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/network-to-amazon-vpc-connectivity-options.html

Connectivity between different VPCs:

If the deployment machine is in AWS, customers must establish the connection between the deployment machine and eks cluster. Since the deployment machine is recommended to be a different VPC than eks cluster, explicitly configuration steps are required to establish the connectivity between two VPC.

Using Transit GW:

Transit GW can be used in hub and spoke mode to attach both the VPCs to transit GW and add routes to respective route tables of private subnets in each VPC pointing to transit GW. Please refer to the below aws documentation about transit gateway configurations.

https://docs.aws.amazon.com/vpc/latest/tgw/tgw-vpc-attachments.html
https://docs.aws.amazon.com/vpc/latest/tgw/tgw-route-tables.html

Custom SSL Certificate for ATOM

To apply a custom SSL certificate or a CA signed to the ATOM we need to follow the below steps.

In case of cloud:

1. Copy the certificate and private key to the jumphost

In case of on prem:

1. Login to the K8s master node as **atom** user
2. Copy the certificate and private key to the k8s master node

Execute following steps.

1. Delete existing secrets
   kubectl delete secret -n atom atom-certificate
   kubectl delete secret -n kube-system tls-secret(only for on-prem)

2. Create new secrets using the new certificate and private key.
   kubectl create secret tls atom-certificate -n atom --cert=<certificate-filename> --key=<private-key-filename> --dry-run -o yaml >cert-atom.yaml
   kubectl create secret tls tls-secret -n kube-system --cert=<certificate-filename> --key=<private-key-filename> --dry-run -o yaml >cert-dashboard.yaml
   NOTE: Replace <certificate-filename> and <private-key-filename> with your files.

3. Apply the secrets
   kubectl create -f cert-atom.yaml -f cert-dashboard.yaml

4. Restart the following pods using *kubectl delete pod -n atom <pod name>*
   a. infra-web-proxy (If certificate change is intended on direct ATOM UI)
   b. keycloak and oauth2_proxy (If certificate change is intended for SSO component)

File Server for ATOM ZTP

AWS EC2

1. Create CentOS 7 EC2 machine
    a. Goto AWS EC2 console
    b. Select Launch Instance, select **CentOS 7 AMI** from the community image
    c. Select the size as **t2.medium**
    d. Select the VPC and subnet that has access to the DHCP server of the required LAB(Target machine of PXE booting and ATOM). Public access can be selected based on access type and need.
    e. Next select the storage of about **100GB**
    f. Next create or select the Security group which provides following access
        i. SSH on TCP/22
        ii. TFTP on UDP/69
        iii. HTTP on TCP/80
        iv. ICMP ping(optional)
    g. Create or select the SSH keypair for accessing the VM
    h. Launch the instance
2. SSH using the private key and run the following command
    a. sudo yum install unzip -y
3. Copy the zip file shared by Anuta into the fileserver host and the extract the file using ***unzip <FILENAME>.zip***
4. Goto the unzipped folder and run ***sudo python pxe_environment.py -a <atom VIP>***
5. Above command should bootstrap the node with required pxeboot files and default structure for the same. **Note:** The default configuration should be overwritten to use.
6. Verify the following files for required values
    a. Update the kickstart file for url and atom.tgz links under **/var/www/html/pxe/ks/centos7-ks.cfg** and make sure that the URL has the connectivity for the target PXE machine.
        i. If kickstart device is not default and requires specific NIC to be mentioned, update the **--device=NIC_NAME** for the network line
    b. Update the PXE config file under **/var/lib/tftpboot/pxelinux.cfg/default**
        i. If kickstart device is not default and requires specific NIC to be mentioned, add **ksdevice=NIC_NAME** at the end of append line
    c. Goto the HTTP file server base path using **cd /var/www/html/pxe**
        i. Extract the default atom.tgz using “***sudo tar -xf atom.tgz***”
        ii. Update the ATOM URL inside using “**sudo vi atom/release/atom.properties**”
        iii. Remove the old atom.tgz using “**sudo rm -f atom.tgz**”
        iv. Recreate the atom.tgz file using “**sudo tar -cf atom.tgz atom**”

ON-PREM VM

- Create a CentOS 7 VM on the VMware or the KVM
    - Provide the specs as 4cpu, 8GB ram and 100GB of storage
    - Provide the static IP and add networking which has connectivity to below ports between ATOM and Fileserver
        - **SSH on TCP/22**
        - **TFTP on UDP/69**
        - **HTTP on TCP/80**
        - ICMP ping(optional)
- SSH using the private key and run the following command
    - sudo yum install unzip -y
- Copy the zip file shared by Anuta into the fileserver host and the extract the file using ***unzip <FILENAME>.zip***
- Goto the unzipped folder(pxeboot) using "cd pxeboot" and run ***sudo python pxe_environment.py -a <atom VIP>***
- Above command should bootstrap the node with required pxeboot files and default structure for the same. **Note:** The default configuration should be overwritten to use.
- Verify the following files for required values
    - Update the kickstart file for url and atom.tgz links under **/var/www/html/pxe/ks/centos7-ks.cfg** and make sure that the URL has the connectivity for the target PXE machine.
    - If kickstart device is not default and requires specific NIC to be mentioned, update the **--device=NIC_NAME** for the network line
    - Update the PXE config file under **/var/lib/tftpboot/pxelinux.cfg/default**
        - If kickstart device is not default and requires specific NIC to be mentioned, add **ksdevice=NIC_NAME** at the end of append line
    - Goto the HTTP file server base path using **cd /var/www/html/pxe**
        - Extract the default atom.tgz using "***sudo tar -xf atom.tgz***"
        - Update the ATOM URL inside using "**sudo vi atom/release/atom.properties**"
        - Remove the old atom.tgz using "**sudo rm -f atom.tgz**"
        - Recreate the atom.tgz file using "**sudo tar -cf atom.tgz atom**"