



ATOM Software Deployment Guide

version 10.6

Table of Contents

Purpose of this document	3
Intended Audience	3
Overview of ATOM Architecture	3
ATOM Deployment	4
Deployment scenarios	4
Local Deployment	4
Distributed Deployment	5
Target Infrastructure	6
On-Prem VMware ESXi, KVM	7
Cloud (GCP / AWS)	7
Requirements	7
Compute, Storage & Memory	7
Basic Non-HA Setup	7
Basic Resilient HA Setup	8
Multi-site Deployment (Remote ATOM agent)	8
ATOM Multi Availability Zone HA Deployment	8
AWS Availability Zones	10
On Premises Across Data Centers / Locations	10
Network Requirements	10
ATOM related Ports/Protocols	10
Kubernetes related Ports/Protocols	11
Linstor related Ports/Protocols	12
IP Addressing Requirements	12
Kubernetes Cluster Requirements	12
Environment Requirements	13
Deployment scripts and files	13
Checklist for managing Kubernetes cluster	13
ATOM Software Requirements	13
Deployment Images	14
Deployment scripts and files	14
Security Apps on VM nodes before ATOM install	15
Procedure for Deploying ATOM	15
New Kubernetes cluster	15
ATOM Deployment	21
Docker registry for Offline deployment	24

ATOM Remote Agent Deployment	39
Procedure of Deploying ATOM in AWS	40
Prerequisites	40
Deploying New K8s Cluster	41
Deploying ATOM	44
ATOM System Manager	45
Post Installation	48
ATOM Single Sign-On (SSO)	49
Google IdP	50
ATOM System Alerts	53
Troubleshooting & FAQ	58
List of useful commands	58
Cleanup of Deployment	59
Guidance on KVM	59
Migration of Storage	60
Steps to check logs in kibana	62
Steps to check load distribution in kafka for config parser	64
Logs for deployment failures	64
Appendix	64
Site-to-Site VPN Setup	64
Create a VPC network	64
Create Customer Gateway	64
Create Virtual Private Gateway	65
Create Site-to-Site VPN Connection	65
Enable Route Propagation	65
Configure Customer Site Gateway	65
Enable ICMP access to AWS EC2 instances(Optional)	65
Custom SSL Certificate for ATOM	66

Purpose of this document

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

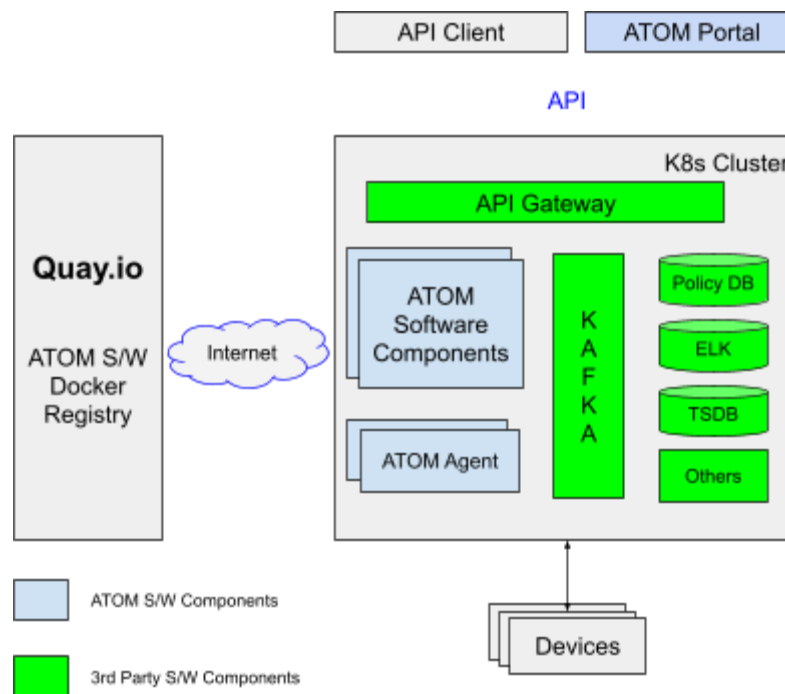
Intended Audience

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.

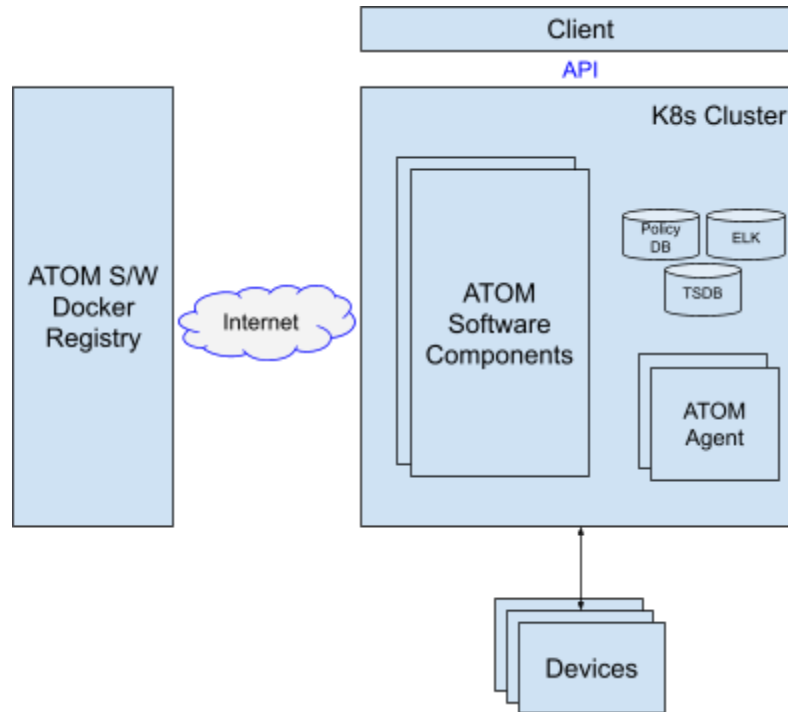
Overview of ATOM Architecture

ATOM software is dockerized and runs on a Kubernetes cluster. In the current release, ATOM is provided as a self-contained installation with all the required components as illustrated below:



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:

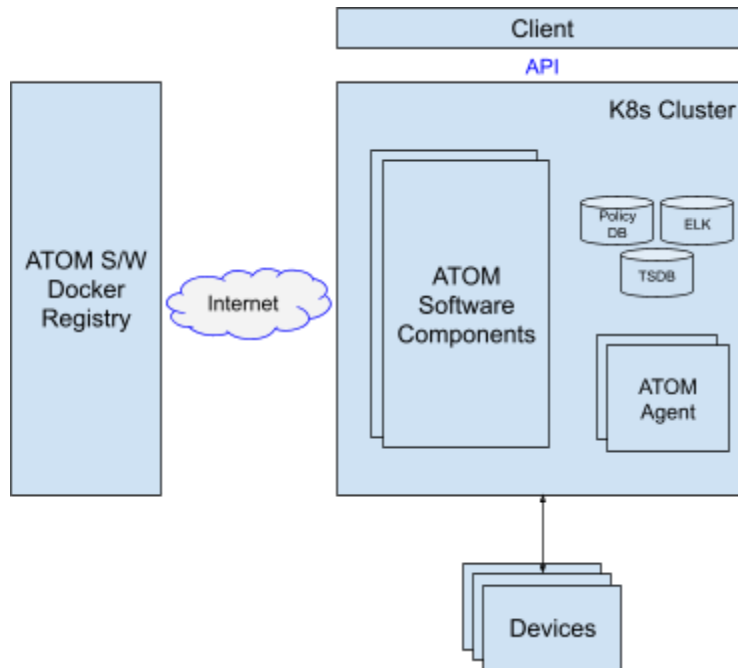
- On-Prem Kubernetes
- Google Cloud Platform (GCP)
- Amazon web service (AWS)

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

- Local Deployment
- Distributed Deployment

Local Deployment

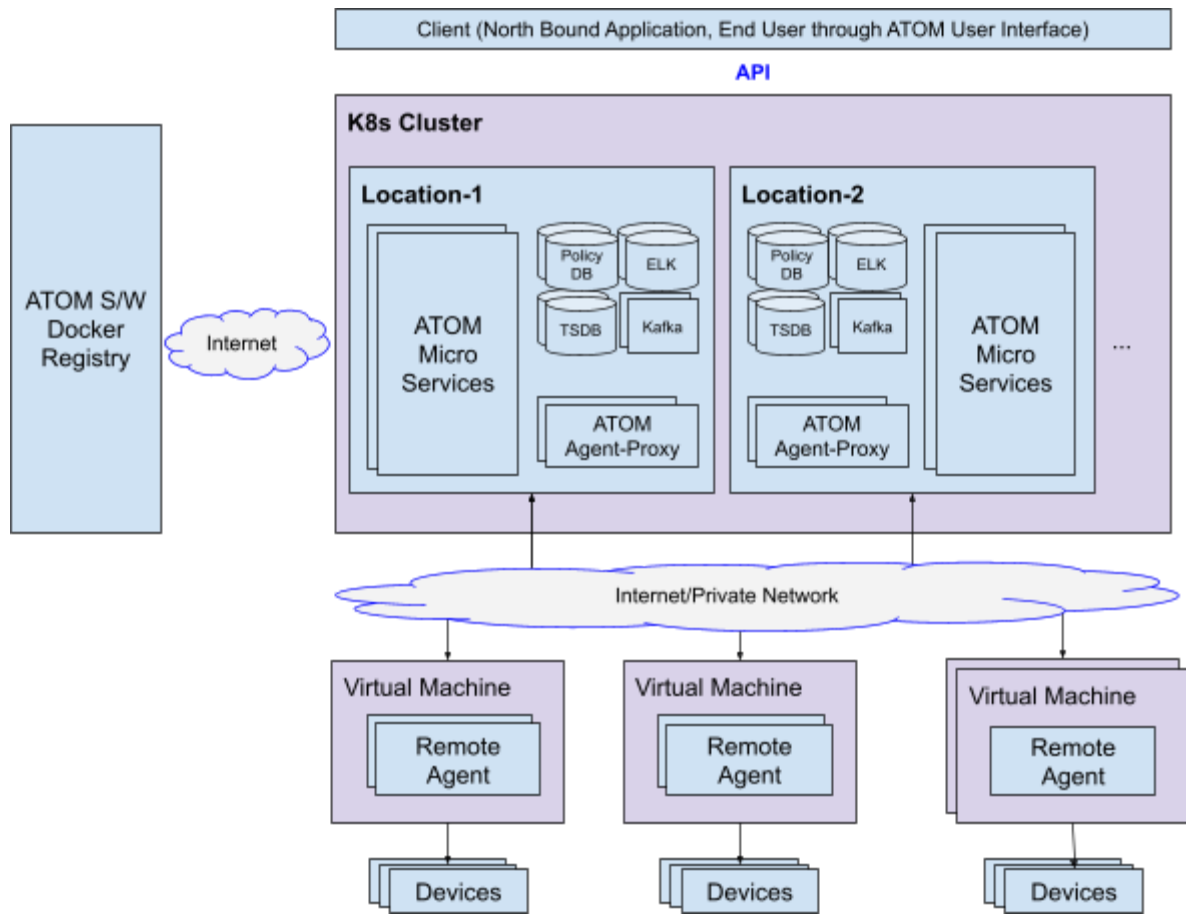
Local deployment has all the ATOM software components deployed in a single Kubernetes cluster.



Distributed Deployment

Distributed deployment allows ATOM software components to be distributed across multiple Kubernetes clusters in the same location or a different geographical location. Distributed Deployment is applicable in the following scenarios:

1. To deploy a Remote Agent - In Some customer scenarios network equipment 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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 can provide 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 etc.

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 *r5.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

Basic Non-HA Setup

A basic Non-HA setup that doesn't support resiliency requires a Kubernetes cluster (1 master and 3 worker nodes) based out of ESXi with recommendations listed below:

Component	Requirements Description
K8s Master - 1 node	Storage reserved in ESXi = 40 GB (SSD) <ul style="list-style-type: none">• CPU - 4 vCPU• Memory - 8GB

K8s Workers - 3 nodes	For each node storage reserved in ESXi = 300 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 32GB
-----------------------	---

Basic Resilient HA Setup

A basic HA setup supporting resiliency with regards to one node or pod failures requires a Kubernetes cluster (3 masters and 7 worker nodes) based out of ESXi with the following details:

Component	Requirements Description
K8s Master - 3 nodes	For each node storage reserved in ESXi = 40 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 8GB
K8s Workers - 7 nodes	For each node storage reserved in ESXi = 300 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 32GB

It is recommended to use RAID10 based storage and provision vms across multiple physical servers.

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 spec is required at each site location(s):

Component	Requirements Description
1 Virtual Machine	Storage reserved in ESXi = 40 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 8GB

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.

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 only from Release-8.8
3. Multi Availability Zones across Regions is yet to be certified in ATOM
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 = 40 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 8GB
K8s Workers - 3 nodes	For each node storage reserved in ESXi = 300 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 32GB

2 Sites Deployment:

Site-1:

Component	Requirements Description
K8s Master - 2 nodes	For each node storage reserved in ESXi = 40 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 8GB
K8s Workers - 4 nodes	For each node storage reserved in ESXi = 300 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 32GB

Site-2:

Component	Requirements Description
K8s Master - 2 nodes	For each node storage reserved in ESXi = 40 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 8GB
K8s Workers - 4 nodes	For each node storage reserved in ESXi = 300 GB (SSD) <ul style="list-style-type: none"> • CPU - 4 vCPU • Memory - 32GB

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]			
ATOM Server (End user Application)	30443	HTTPs access	This will be the ATOM GUI page served via HAProxy.
Single Sign-On	32443	HTTPs access	For Single Sign-On Login access for ATOM, Grafana, Kibana, Glowroot and Kafka-Manager.
Alert Manager	31090, 31093, 32090	HTTP access	Monitoring/Debugging Alerts
Minio	31311	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 / Nodeport (30700)	TCP/RSocket	Remote Agent communicates with Agent-proxy via agent-lb.

Southbound communication with network devices			
ATOM Agent - Network Elements	23	Telnet to network devices (TCP)	Different ports are used for various usecases in ATOM. Make sure PING reachability is also there.
	21	FTP to network device (TCP)	
	22	SSH to network devices (TCP)	
	161	SNMP to network devices (UDP)	
	162	SNMP Trap Listening (Server) from network devices (UDP)	
	69	TFTP to network devices (UDP)	
	514	SYSLOG Listening (Server) port from network devices (UDP)	
	830	NETCONF to network devices (TCP)	
	32222	SFTP from ATOM Server to devices if ATOM needs to act as Image Server (TCP)	

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.

ATOM -> Required Public Access Details for Firewall if applicable.	Port/Protocol	Domain Name
	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

Kubernetes related Ports/Protocols

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

ATOM uses linstor CSI driver as a storage provisioner. Below are the Ports and Protocols which 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 HA Master Resilient setup, when 3 Masters is opted, reserve one extra IP(virtual IP) belonging to the same subnet as other 3 Masters.
- Internally for communication between microservices we use 10.200.0.0/16 subnet. If this subnet conflicts with any of the lab device subnet IPs then it needs to be mentioned beforehand and it can be handled accordingly during ATOM deployment.

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>) on CentOS hosts

Any Other variations of Kubernetes Distribution requires additional validation from Anuta and requires significant lead time depending on the distribution.

Anuta provides deployment artifacts such as OVA images for CentOS vms, scripts for creating the Kubernetes cluster and 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. OVA (Centos server with pre-installed minimal packages) is already imported manually into vCenter template library. OVAs should have been obtained from Anuta

For bootstrapping the Kubernetes cluster, scripts obtained from Anuta when triggered will help in installations of below packages and subsequent ATOM deployment. Refer to the section, [“Procedure for Deploying ATOM”](#).

- Docker-ce v20.10.5
- Kubectl v1.19.8
- Helm v3.3

Environment Requirements

Anuta scripts requirements for Kubernetes cluster creation are as follows -

1. Static IP addresses with unique hostnames should be provided manually (recommended than DHCP).
2. ESXi/vCenter version to be 6.0+ and OVA [Centos server with pre-installed minimal packages (shared by Anuta)] is imported into any of vCenter hosts

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.

Checklist for managing Kubernetes cluster

If you have already Kubernetes cluster created, ensure that the following are in place before deploying the ATOM application.

1. kubectl is installed and Kubernetes cluster is up and running
2. Helm is installed. https://docs.helm.sh/using_helm/#install

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:8.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 kube, 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

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

ATOM applications can be deployed on new Kubernetes with help of [Deployment scripts and files](#) provided by Anuta.

New Kubernetes cluster

As discussed in the section "[Environment Requirements](#)", provide the static IP address for VMs, and follow the sequence of steps for bootstrapping the Kubernetes cluster.

1. Verify that you have imported the shared Anuta OVA templates into your VMware vCenter.
2. For the minimal setup, create 4 VMs (1 master and 3 worker nodes) using Centos OVA template provided above by Anuta. Similar approach can be done for Basic Resilient HA setup.
 - a. The specs for master node will be 4CPU/8GB RAM/40 SSD/1 NIC
 - b. Specs for each worker node will be 4CPU/32GB RAM/300GB SSD/1 NIC

- c. Login credentials for these VMs 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
3. 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]:

```
[atom@scmchange-deploy-master-9780 ~]$ pwd
/home/atom
[atom@scmchange-deploy-master-9780 ~]$ ll | grep node_setup
-rw-rw-r-- 1 atom atom 20439 Mar 31 13:05 node_setup.py
[atom@scmchange-deploy-master-9780 ~]$
```

4. 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.
5. 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
```

6. 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      0 BMU
ens160     1500     9161      0    7923  0          498      0      0      0      0 BMRU
lo         65536     208      0      0      0          208      0      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]

7. Make sure Internet access is there from all the nodes.
8. After completion of the bootstrap process we are now ready to begin the atom installation process. Select Atom Installation on the Master Node and proceed.

```

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

```

9. Since it is a fresh install we can enter choice 1 and begin the process. If k8s cluster is already set up in the customer lab he can proceed with Atom software download and installation by selecting appropriate choices.
10. To download the **Atom Software** 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/0ko67lke0c48nx1/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
```

11. After the 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:
For Basic Non HA setup follow below

```

!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Do you want Minimal or Resilient setup?(M/R)m

Enter the Master IP [Info :Virtual IP in case of Resilient setup]:172.16.19.20
wrapper properties updated
Master-0
Enter the IP:172.16.19.20
Worker-0
Enter the IP:172.16.19.21
Worker-1
Enter the IP:172.16.19.22
Worker-2
Enter the IP:172.16.19.23
Master Node list is:
['172.16.19.20']
Worker Node list is :
['172.16.19.21', '172.16.19.22', '172.16.19.23']
Build number is: 10.5.0.0.40678

```

For Resilient HA setup follow below

```

Do you want Minimal or Resilient setup?(M/R)R

Enter the number of master nodes[Minimum value : 3]:3
Enter the number of worker nodes[Minimum value : 7]:7
Reserve an extra IP to be used as Virtual IP in case of Resilient Setup
Enter the Master IP [Info :Virtual IP in case of Resilient setup]:172.16.19.50

```

```

Master-0
Enter the IP:172.16.19.40
Master-1
Enter the IP:172.16.19.41
Master-2
Enter the IP:172.16.19.42
Worker-0
Enter the IP:172.16.19.43
Worker-1
Enter the IP:172.16.19.44
Worker-2
Enter the IP:172.16.19.45
Worker-3
Enter the IP:172.16.19.46
Worker-4
Enter the IP:172.16.19.47
Worker-5
Enter the IP:172.16.19.48
Worker-6
Enter the IP:172.16.19.49
Master Node list is:
['172.16.19.40', '172.16.19.41', '172.16.19.42']
Worker Node list is :
['172.16.19.43', '172.16.19.44', '172.16.19.45', '172.16.19.46', '172.16.19.47', '172.16.19.48', '172.16.19.49']
Build number is: 10.5.0.0.40791

```

12. Above script can create K8s cluster among Master and Worker Nodes spread across Esxi/Locations which have reachability.
13. Verify the node creation using the command **"kubectl get nodes"** and verify labels using the command **"kubectl get nodes --show-labels"**
14. As the Kubernetes cluster's bootstrapping is already done, it is ready for ATOM deployment. Follow the steps outlined in the section, "[ATOM Deployment](#)" to complete the ATOM deployment process.

ATOM Deployment

After ensuring that the prerequisites are taken care as described in the section, "[Prerequisites for Deploying ATOM](#)" and perform the following steps:

1. For Non-HA or Resilient HA setup, ensure that the worker nodes are labelled properly as below.

For Non-HA setup:

Worker 1:

elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb

Worker 2:

elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb

Worker 3:

elasticsearch,broker,zookeeper,object_store,default_agent,grafana,distributed_db,agent1,securestore,northbound,thanos,monitoring_server,infra-tsdb

For Resilient-HA setup:

Worker 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

Worker 2:

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

Worker 3:

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

Worker 4:

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

Worker 5:

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

Worker 6:

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

You can view the existing labels using **kubectl get nodes --show-labels**. To label any node with a specific label then use below command:

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

Note: Make sure you label dc1, dc2 appropriately based on the datacenter where it is present. For more Worker Nodes also the labelling approach remains the same as above.

2. To download the **Atom Software** provided by Anuta Networks Support team we can use any of the methods as described in the section [New Kubernetes cluster](#) step number 10.

3. On the master node of the Kubernetes cluster, you can deploy the components of the application by selecting the Atom installation option.

Python script “deploy_atom.py” will execute and use the values.yaml file which has values set for various parameters and the images to be used for deployment.

NOTE: The order in which these components should be deployed is already defined in the above script.

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

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

namespace: <mynamespace>
```

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

```
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
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-file-server 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-frontend got successfully deployed
Helm chart infra-glowroot got successfully deployed
Helm chart burrow got successfully deployed
Helm chart es-curator 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

```

4. To get a summary of access URLs for various components deployed, you can execute following command in scripts folder:

```

# cd scripts
# sh get_urls.sh

```

The output will be similar to below

```

atom URL : https://172.16.22.207:30443/
atom-kpi-metrics URL : http://172.16.22.207:/
Error from server (NotFound): services "infra-tsdb-query" not found
service domain-metrics is not deployed
service kafka-manager is not deployed
SSO URLs for application endpoints are:
ATOM UI ==> https://172.16.20.241:32443
KEYCLOAK UI ==> https://172.16.20.241:32443/auth
KIBANA UI ==> https://172.16.20.241:32443/kibana
GRAFANA UI ==> https://172.16.20.241:32443/grafana
GLOWROOT UI ==> https://172.16.20.241:32443/glowroot
K8S UI ==> https://172.16.20.241:32443/k8s/
KAFKA MANAGER UI ==> invalid or not deployed
Fetching token for Kubernetes Dashboard login
eyJhbGciOiJSUzI1NiIsImtpZCI6ImlkSlJIVHVfaHZsc3NndUM4NDRFMmdqU1FLTlVWekh
OcXZOUGFNNDExZFEifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3V
iZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYWw1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZX
JuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImNsdXN0ZXItYWRtaW4tZ
GFzaGJvYXJkLXNhLXRva2VuLXRjdGY3Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3V
dC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImNsdXN0ZXItYWRtaW4tZGFzaGJvYXJkLXNhIiw

```



```
ia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiY2
EzNTM0ZDMtYmE2YS00ZDaxLWE2YTItYmRiZmUyZTYwMDk0Iiwic3ViIjoic3lzdGVtOnNlc
nZpY2VhY2NvdW50OmRlZmFlbHQ6Y2xlc3RlcilhZG1pbi1kYXNoYm9hcmQtc2EifQ.yWSYa
xanwGtd3HdQTVRzv8KBfi_EPxqs0yruKEM5rxaXZdW6fdQBhD1oNtxWoUN12UUOLhB_m82Q
hWRwZlaXCqBELPjSe96QJlH68e-1mlghJXOGm2uiFE251Lcdxu0bwNN73Xw26h9D4Tz1EzQ
6_oasBrTJP3J479ik8M7U1LmtktRor9xpFV7VJb6DDbDjhycJTnAkuFZD6Walmjj2NE_g1q
kfAQstS2B6vojZvsPEbyltO6lOCW7YCuIg_kaAhakR22nbUXg2OftNgCogBlSmzf8Gw3S1t
EA89tuZt-mk_ugpaHayCdxwSXpUR-MPGGIvngyUX7IeY3rEMwqNYQ
```

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.

The registry template “docker-registry-300-0421” is the OVA that can be used to deploy the Deploy the OVA through 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.tpl
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
20fle7953be4: 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
1ef3f81elf85: 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
6e7fceeef62a: 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
daleb793d5c9: 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>”.

```

[atom@docker ~]$ sudo python docker-registry.py -r 172.16.26.105 -p repo -t
push -v 10.4.0.0.40620
[sudo] password for atom:
INFO: Debug logs are sent to atom-registry.log
INFO: ['docker-registry.py', '-r', '172.16.26.105', '-p', 'repo', '-t',
'push', '-v', '10.4.0.0.40620']
INFO: Push task is selected
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:v1.19.8.tar
INFO: docker load -qi images/kubernetes/kube-apiserver:v1.19.8.tar
INFO: docker load -qi images/kubernetes/coredns:1.7.0.tar
INFO: docker load -qi images/kubernetes/kube-scheduler:v1.19.8.tar
INFO: docker load -qi images/kubernetes/kube-controller-manager:v1.19.8.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:3.2.tar
INFO: docker tag k8s.gcr.io/etcd:3.4.13-0 172.16.26.105/repo/etcd:3.4.13-0
INFO: docker tag k8s.gcr.io/kube-proxy:v1.19.8
172.16.26.105/repo/kube-proxy:v1.19.8
INFO: docker tag k8s.gcr.io/kube-apiserver:v1.19.8
172.16.26.105/repo/kube-apiserver:v1.19.8
INFO: docker tag k8s.gcr.io/coredns:1.7.0 172.16.26.105/repo/coredns:1.7.0
INFO: docker tag k8s.gcr.io/kube-scheduler:v1.19.8
172.16.26.105/repo/kube-scheduler:v1.19.8
INFO: docker tag k8s.gcr.io/kube-controller-manager:v1.19.8
172.16.26.105/repo/kube-controller-manager:v1.19.8
INFO: docker tag k8s.gcr.io/metrics-server/metrics-server:v0.4.2
172.16.26.105/repo/metrics-server:v0.4.2
INFO: docker tag gcr.io/heptio-images/eventrouter:v0.3
172.16.26.105/repo/eventrouter:v0.3
INFO: docker tag k8s.gcr.io/pause:3.2 172.16.26.105/repo/pause:3.2
INFO: docker push 172.16.26.105/repo/etcd:3.4.13-0
INFO: docker push 172.16.26.105/repo/kube-proxy:v1.19.8
INFO: docker push 172.16.26.105/repo/kube-apiserver:v1.19.8
INFO: docker push 172.16.26.105/repo/coredns:1.7.0
INFO: docker push 172.16.26.105/repo/kube-scheduler:v1.19.8
INFO: docker push 172.16.26.105/repo/kube-controller-manager:v1.19.8

```



```

INFO: docker push 172.16.26.105/repo/metrics-server:v0.4.2
INFO: docker push 172.16.26.105/repo/eventrouter:v0.3
INFO: docker push 172.16.26.105/repo/pause:3.2
INFO: docker image prune -af
INFO: tar -xf images/standalone.tgz
INFO: docker load -qi images/standalone/modsecurity-spoa:v0.6.tar
INFO: docker load -qi images/standalone/jaeger-cassandra-schema:1.19.2.tar
INFO: docker load -qi images/standalone/alertmanager:v0.21.0.tar
INFO: docker load -qi images/standalone/thanos:v0.14.0.tar
INFO: docker load -qi images/standalone/jaeger-collector:1.19.2.tar
INFO: docker load -qi images/standalone/prometheus:v2.20.1.tar
INFO: docker load -qi images/standalone/busybox:1.28.tar
INFO: docker load -qi images/standalone/rsyslog:latest.tar
INFO: docker load -qi images/standalone/thanos-receive-controller:latest.tar
INFO: docker load -qi images/standalone/example-hotrod:1.19.2.tar
INFO: docker load -qi
images/standalone/minio:RELEASE.2020-07-27T18-37-02Z.tar
INFO: docker load -qi images/standalone/configmap-reload:v0.3.0.tar
INFO: docker load -qi images/standalone/mc:RELEASE.2020-07-17T02-52-20Z.tar
INFO: docker load -qi images/standalone/helm-kubect1-jq:3.1.0.tar
INFO: docker load -qi images/standalone/pushgateway:v0.8.0.tar
INFO: docker load -qi images/standalone/node-exporter:v1.0.1.tar
INFO: docker load -qi images/standalone/jaeger-query:1.19.2.tar
INFO: docker load -qi images/standalone/kube-state-metrics:v1.9.5.tar
INFO: docker load -qi images/standalone/jaeger-agent:1.19.2.tar
INFO: docker load -qi images/standalone/spark-dependencies:latest.tar
INFO: docker tag quay.io/jcmoraisjr/modsecurity-spoa:v0.6
172.16.26.105/repo/modsecurity-spoa:v0.6
INFO: docker tag jaegertracing/jaeger-cassandra-schema:1.19.2
172.16.26.105/repo/jaeger-cassandra-schema:1.19.2
INFO: docker tag prom/alertmanager:v0.21.0
172.16.26.105/repo/alertmanager:v0.21.0
INFO: docker tag quay.io/thanos/thanos:v0.14.0
172.16.26.105/repo/thanos:v0.14.0
INFO: docker tag jaegertracing/jaeger-collector:1.19.2
172.16.26.105/repo/jaeger-collector:1.19.2
INFO: docker tag prom/prometheus:v2.20.1
172.16.26.105/repo/prometheus:v2.20.1
INFO: docker tag library/busybox:1.28 172.16.26.105/repo/busybox:1.28
INFO: docker tag jumanjiman/rsyslog:latest 172.16.26.105/repo/rsyslog:latest
INFO: docker tag quay.io/observatorium/thanos-receive-controller:latest
172.16.26.105/repo/thanos-receive-controller:latest
INFO: docker tag jaegertracing/example-hotrod:1.19.2
172.16.26.105/repo/example-hotrod:1.19.2
INFO: docker tag minio/minio:RELEASE.2020-07-27T18-37-02Z
172.16.26.105/repo/minio:RELEASE.2020-07-27T18-37-02Z
INFO: docker tag jimmidyson/configmap-reload:v0.3.0
172.16.26.105/repo/configmap-reload:v0.3.0
INFO: docker tag minio/mc:RELEASE.2020-07-17T02-52-20Z
172.16.26.105/repo/mc:RELEASE.2020-07-17T02-52-20Z
INFO: docker tag bskim45/helm-kubect1-jq:3.1.0
172.16.26.105/repo/helm-kubect1-jq:3.1.0
INFO: docker tag prom/pushgateway:v0.8.0
172.16.26.105/repo/pushgateway:v0.8.0
INFO: docker tag prom/node-exporter:v1.0.1
172.16.26.105/repo/node-exporter:v1.0.1

```

```

INFO: docker tag jaegertracing/jaeger-query:1.19.2
172.16.26.105/repo/jaeger-query:1.19.2
INFO: docker tag quay.io/coreos/kube-state-metrics:v1.9.5
172.16.26.105/repo/kube-state-metrics:v1.9.5
INFO: docker tag jaegertracing/jaeger-agent:1.19.2
172.16.26.105/repo/jaeger-agent:1.19.2
INFO: docker tag jaegertracing/spark-dependencies:latest
172.16.26.105/repo/spark-dependencies:latest
INFO: docker push 172.16.26.105/repo/modsecurity-spoa:v0.6
INFO: docker push 172.16.26.105/repo/jaeger-cassandra-schema:1.19.2
INFO: docker push 172.16.26.105/repo/alertmanager:v0.21.0
INFO: docker push 172.16.26.105/repo/thanos:v0.14.0
INFO: docker push 172.16.26.105/repo/jaeger-collector:1.19.2
INFO: docker push 172.16.26.105/repo/prometheus:v2.20.1
INFO: docker push 172.16.26.105/repo/busybox:1.28
INFO: docker push 172.16.26.105/repo/rsyslog:latest
INFO: docker push 172.16.26.105/repo/thanos-receive-controller:latest
INFO: docker push 172.16.26.105/repo/example-hotrod:1.19.2
INFO: docker push 172.16.26.105/repo/minio:RELEASE.2020-07-27T18-37-02Z
INFO: docker push 172.16.26.105/repo/configmap-reload:v0.3.0
INFO: docker push 172.16.26.105/repo/mc:RELEASE.2020-07-17T02-52-20Z
INFO: docker push 172.16.26.105/repo/helm-kubect1-jq:3.1.0
INFO: docker push 172.16.26.105/repo/pushgateway:v0.8.0
INFO: docker push 172.16.26.105/repo/node-exporter:v1.0.1
INFO: docker push 172.16.26.105/repo/jaeger-query:1.19.2
INFO: docker push 172.16.26.105/repo/kube-state-metrics:v1.9.5
INFO: docker push 172.16.26.105/repo/jaeger-agent:1.19.2
INFO: docker push 172.16.26.105/repo/spark-dependencies:latest
INFO: docker image prune -af
INFO: tar -xf images/databases.tgz
INFO: docker load -qi images/databases/infra-broker:6.0.0.08102020.tar
INFO: docker load -qi images/databases/infra-elasticsearch:770.tar
INFO: docker load -qi images/databases/cp-schema-registry:6.0.0.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:0.3.tar
INFO: docker load -qi images/databases/postgres_exporter:v0.8.0.tar
INFO: docker load -qi images/databases/infra-postgres:11.2.tar
INFO: docker load -qi images/databases/elasticsearch-config:0.1.tar
INFO: docker load -qi images/databases/kafka-operator:0.4.tar
INFO: docker load -qi images/databases/infra-logstash:772_150321_v2.tar
INFO: docker load -qi images/databases/infra-schema_repo_init:1.0.tar
INFO: docker load -qi images/databases/infra-zookeeper:6.0.0.08102020.tar
INFO: docker load -qi images/databases/infra-distributed-db:8.7.29.tar
INFO: docker tag quay.io/release/infra-broker:6.0.0.08102020
172.16.26.105/repo/infra-broker:6.0.0.08102020
INFO: docker tag quay.io/release/infra-elasticsearch:770
172.16.26.105/repo/infra-elasticsearch:770
INFO: docker tag confluentinc/cp-schema-registry:6.0.0
172.16.26.105/repo/cp-schema-registry:6.0.0
INFO: docker tag quay.io/release/infra-filebeat:770
172.16.26.105/repo/infra-filebeat:770

```

```
INFO: docker tag quay.io/release/infra-distributed-db-webconsole:0.2
172.16.26.105/repo/infra-distributed-db-webconsole:0.2
INFO: docker tag quay.io/release/infra-kafka-prometheus-jmx-exporter:0.11.0
172.16.26.105/repo/infra-kafka-prometheus-jmx-exporter:0.11.0
INFO: docker tag quay.io/release/infra-distributed-db-webagent:0.3
172.16.26.105/repo/infra-distributed-db-webagent:0.3
INFO: docker tag wrouesnel/postgres_exporter:v0.8.0
172.16.26.105/repo/postgres_exporter:v0.8.0
INFO: docker tag quay.io/release/infra-postgres:11.2
172.16.26.105/repo/infra-postgres:11.2
INFO: docker tag quay.io/release/elasticsearch-config:0.1
172.16.26.105/repo/elasticsearch-config:0.1
INFO: docker tag quay.io/release/kafka-operator:0.4
172.16.26.105/repo/kafka-operator:0.4
INFO: docker tag quay.io/release/infra-logstash:772_150321_v2
172.16.26.105/repo/infra-logstash:772_150321_v2
INFO: docker tag quay.io/release/infra-schema_repo_init:1.0
172.16.26.105/repo/infra-schema_repo_init:1.0
INFO: docker tag quay.io/release/infra-zookeeper:6.0.0.08102020
172.16.26.105/repo/infra-zookeeper:6.0.0.08102020
INFO: docker tag quay.io/release/infra-distributed-db:8.7.29
172.16.26.105/repo/infra-distributed-db:8.7.29
INFO: docker push 172.16.26.105/repo/infra-broker:6.0.0.08102020
INFO: docker push 172.16.26.105/repo/infra-elasticsearch:770
INFO: docker push 172.16.26.105/repo/cp-schema-registry:6.0.0
INFO: docker push 172.16.26.105/repo/infra-filebeat:770
INFO: docker push 172.16.26.105/repo/infra-distributed-db-webconsole:0.2
INFO: docker push
172.16.26.105/repo/infra-kafka-prometheus-jmx-exporter:0.11.0
INFO: docker push 172.16.26.105/repo/infra-distributed-db-webagent:0.3
INFO: docker push 172.16.26.105/repo/postgres_exporter:v0.8.0
INFO: docker push 172.16.26.105/repo/infra-postgres:11.2
INFO: docker push 172.16.26.105/repo/elasticsearch-config:0.1
INFO: docker push 172.16.26.105/repo/kafka-operator:0.4
INFO: docker push 172.16.26.105/repo/infra-logstash:772_150321_v2
INFO: docker push 172.16.26.105/repo/infra-schema_repo_init:1.0
INFO: docker push 172.16.26.105/repo/infra-zookeeper:6.0.0.08102020
INFO: docker push 172.16.26.105/repo/infra-distributed-db:8.7.29
INFO: docker image prune -af
INFO: tar -xf images/infra.tgz
INFO: docker load -qi images/infra/infra-glowroot:0.8.tar
INFO: docker load -qi images/infra/curator:5.7.6.tar
INFO: docker load -qi images/infra/haproxy-ingress:v0.11.tar
INFO: docker load -qi images/infra/infra-kafka-control:0.2.0.tar
INFO: docker load -qi images/infra/burrow-exporter:latest.tar
INFO: docker load -qi images/infra/infra-burrow:1.2.2.tar
INFO: docker load -qi images/infra/kafka-manager:latest.tar
INFO: docker tag quay.io/release/infra-glowroot:0.8
172.16.26.105/repo/infra-glowroot:0.8
INFO: docker tag bobrik/curator:5.7.6 172.16.26.105/repo/curator:5.7.6
INFO: docker tag quay.io/jcmoraisjr/haproxy-ingress:v0.11
172.16.26.105/repo/haproxy-ingress:v0.11
INFO: docker tag quay.io/release/infra-kafka-control:0.2.0
172.16.26.105/repo/infra-kafka-control:0.2.0
INFO: docker tag solsson/burrow-exporter:latest
172.16.26.105/repo/burrow-exporter:latest
```

```

INFO: docker tag quay.io/release/infra-burrow:1.2.2
172.16.26.105/repo/infra-burrow:1.2.2
INFO: docker tag solsson/kafka-manager:latest
172.16.26.105/repo/kafka-manager:latest
INFO: docker push 172.16.26.105/repo/infra-glowroot:0.8
INFO: docker push 172.16.26.105/repo/curator:5.7.6
INFO: docker push 172.16.26.105/repo/haproxy-ingress:v0.11
INFO: docker push 172.16.26.105/repo/infra-kafka-control:0.2.0
INFO: docker push 172.16.26.105/repo/burrow-exporter:latest
INFO: docker push 172.16.26.105/repo/infra-burrow:1.2.2
INFO: docker push 172.16.26.105/repo/kafka-manager:latest
INFO: docker image prune -af
INFO: tar -xf images/sso.tgz
INFO: docker load -qi images/sso/keycloak:12.0.1-1.tar
INFO: docker load -qi images/sso/infra-grafana:7.0.0.0v4.tar
INFO: docker load -qi images/sso/dashboard:v2.0.0-rc5.tar
INFO: docker load -qi images/sso/oauth2-proxy:v7.0.1-verifyjwt.tar
INFO: docker load -qi images/sso/infra-kibana:770.tar
INFO: docker load -qi images/sso/metrics-scraper:v1.0.3.tar
INFO: docker load -qi images/sso/echoserver:1.3.tar
INFO: docker tag quay.io/release/keycloak:12.0.1-1
172.16.26.105/repo/keycloak:12.0.1-1
INFO: docker tag quay.io/release/infra-grafana:7.0.0.0v4
172.16.26.105/repo/infra-grafana:7.0.0.0v4
INFO: docker tag kubernetesui/dashboard:v2.0.0-rc5
172.16.26.105/repo/dashboard:v2.0.0-rc5
INFO: docker tag quay.io/release/oauth2-proxy:v7.0.1-verifyjwt
172.16.26.105/repo/oauth2-proxy:v7.0.1-verifyjwt
INFO: docker tag quay.io/release/infra-kibana:770
172.16.26.105/repo/infra-kibana:770
INFO: docker tag kubernetesui/metrics-scraper:v1.0.3
172.16.26.105/repo/metrics-scraper:v1.0.3
INFO: docker tag gcr.io/google_containers/echoserver:1.3
172.16.26.105/repo/echoserver:1.3
INFO: docker push 172.16.26.105/repo/keycloak:12.0.1-1
INFO: docker push 172.16.26.105/repo/infra-grafana:7.0.0.0v4
INFO: docker push 172.16.26.105/repo/dashboard:v2.0.0-rc5
INFO: docker push 172.16.26.105/repo/oauth2-proxy:v7.0.1-verifyjwt
INFO: docker push 172.16.26.105/repo/infra-kibana:770
INFO: docker push 172.16.26.105/repo/metrics-scraper:v1.0.3
INFO: docker push 172.16.26.105/repo/echoserver:1.3
INFO: docker image prune -af
INFO: tar -xf images/linstor.tgz
INFO: docker load -qi images/linstor/linstor-operator:v1.4.0.tar
INFO: docker load -qi images/linstor/csi-provisioner:v2.1.1.tar
INFO: docker load -qi images/linstor/linstor-k8s-ha-controller:v0.1.3.tar
INFO: docker load -qi images/linstor/centos:8.tar
INFO: docker load -qi images/linstor/csi-attacher:v3.1.0.tar
INFO: docker load -qi images/linstor/linstor-satellite:v1.11.1.tar
INFO: docker load -qi images/linstor/csi-snapshotter:v3.0.3.tar
INFO: docker load -qi images/linstor/kube-scheduler-amd64:v1.18.6.tar
INFO: docker load -qi images/linstor/livenessprobe:v2.2.0.tar
INFO: docker load -qi images/linstor/snapshot-controller:v3.0.3.tar
INFO: docker load -qi images/linstor/linstor-controller:v1.11.1.tar
INFO: docker load -qi images/linstor/linstor-csi:v0.12.1.tar
INFO: docker load -qi images/linstor/etcd:v3.4.15.tar

```

```

INFO: docker load -qi images/linstor/drbd9-rhel7:v9.0.28.tar
INFO: docker load -qi images/linstor/csi-node-driver-registrar:v2.1.0.tar
INFO: docker load -qi images/linstor/stork:2.6.2.tar
INFO: docker load -qi images/linstor/csi-resizer:v1.1.0.tar
INFO: docker tag quay.io/release/linstor-operator:v1.4.0
172.16.26.105/repo/linstor-operator:v1.4.0
INFO: docker tag k8s.gcr.io/sig-storage/csi-provisioner:v2.1.1
172.16.26.105/repo/csi-provisioner:v2.1.1
INFO: docker tag quay.io/release/linstor-k8s-ha-controller:v0.1.3
172.16.26.105/repo/linstor-k8s-ha-controller:v0.1.3
INFO: docker tag quay.io/centos/centos:8 172.16.26.105/repo/centos:8
INFO: docker tag k8s.gcr.io/sig-storage/csi-attacher:v3.1.0
172.16.26.105/repo/csi-attacher:v3.1.0
INFO: docker tag quay.io/release/linstor-satellite:v1.11.1
172.16.26.105/repo/linstor-satellite:v1.11.1
INFO: docker tag k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3
172.16.26.105/repo/csi-snapshotter:v3.0.3
INFO: docker tag k8s.gcr.io/kube-scheduler-amd64:v1.18.6
172.16.26.105/repo/kube-scheduler-amd64:v1.18.6
INFO: docker tag k8s.gcr.io/sig-storage/livenessprobe:v2.2.0
172.16.26.105/repo/livenessprobe:v2.2.0
INFO: docker tag k8s.gcr.io/sig-storage/snapshot-controller:v3.0.3
172.16.26.105/repo/snapshot-controller:v3.0.3
INFO: docker tag quay.io/release/linstor-controller:v1.11.1
172.16.26.105/repo/linstor-controller:v1.11.1
INFO: docker tag quay.io/release/linstor-csi:v0.12.1
172.16.26.105/repo/linstor-csi:v0.12.1
INFO: docker tag gcr.io/etcd-development/etcd:v3.4.15
172.16.26.105/repo/etcd:v3.4.15
INFO: docker tag quay.io/release/drbd9-rhel7:v9.0.28
172.16.26.105/repo/drbd9-rhel7:v9.0.28
INFO: docker tag k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.1.0
172.16.26.105/repo/csi-node-driver-registrar:v2.1.0
INFO: docker tag docker.io/openstorage/stork:2.6.2
172.16.26.105/repo/stork:2.6.2
INFO: docker tag k8s.gcr.io/sig-storage/csi-resizer:v1.1.0
172.16.26.105/repo/csi-resizer:v1.1.0
INFO: docker push 172.16.26.105/repo/linstor-operator:v1.4.0
INFO: docker push 172.16.26.105/repo/csi-provisioner:v2.1.1
INFO: docker push 172.16.26.105/repo/linstor-k8s-ha-controller:v0.1.3
INFO: docker push 172.16.26.105/repo/centos:8
INFO: docker push 172.16.26.105/repo/csi-attacher:v3.1.0
INFO: docker push 172.16.26.105/repo/linstor-satellite:v1.11.1
INFO: docker push 172.16.26.105/repo/csi-snapshotter:v3.0.3
INFO: docker push 172.16.26.105/repo/kube-scheduler-amd64:v1.18.6
INFO: docker push 172.16.26.105/repo/livenessprobe:v2.2.0
INFO: docker push 172.16.26.105/repo/snapshot-controller:v3.0.3
INFO: docker push 172.16.26.105/repo/linstor-controller:v1.11.1
INFO: docker push 172.16.26.105/repo/linstor-csi:v0.12.1
INFO: docker push 172.16.26.105/repo/etcd:v3.4.15
INFO: docker push 172.16.26.105/repo/drbd9-rhel7:v9.0.28
INFO: docker push 172.16.26.105/repo/csi-node-driver-registrar:v2.1.0
INFO: docker push 172.16.26.105/repo/stork:2.6.2
INFO: docker push 172.16.26.105/repo/csi-resizer:v1.1.0
INFO: docker image prune -af
INFO: tar -xf images/atom.tgz

```

```

INFO: docker load -qi images/atom/atom-agent-proxy:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-telemetry-engine:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-pnp-server:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-file-server:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-workflow-engine:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-isim:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-inventory-mgr:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-core:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-agent:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-telemetry-exporter:10.4.0.0.40620.tar
INFO: docker load -qi images/atom/atom-scheduler:10.4.0.0.40620.tar
INFO: docker tag quay.io/release/atom-agent-proxy:10.4.0.0.40620
172.16.26.105/repo/atom-agent-proxy:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-telemetry-engine:10.4.0.0.40620
172.16.26.105/repo/atom-telemetry-engine:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-pnp-server:10.4.0.0.40620
172.16.26.105/repo/atom-pnp-server:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-file-server:10.4.0.0.40620
172.16.26.105/repo/atom-file-server:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-workflow-engine:10.4.0.0.40620
172.16.26.105/repo/atom-workflow-engine:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-isim:10.4.0.0.40620
172.16.26.105/repo/atom-isim:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-inventory-mgr:10.4.0.0.40620
172.16.26.105/repo/atom-inventory-mgr:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-core:10.4.0.0.40620
172.16.26.105/repo/atom-core:10.4.0.0.40620
INFO: docker tag quay.io/atom-agent/atom-agent:10.4.0.0.40620
172.16.26.105/repo/atom-agent:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-telemetry-exporter:10.4.0.0.40620
172.16.26.105/repo/atom-telemetry-exporter:10.4.0.0.40620
INFO: docker tag quay.io/release/atom-scheduler:10.4.0.0.40620
172.16.26.105/repo/atom-scheduler:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-agent-proxy:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-telemetry-engine:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-pnp-server:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-file-server:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-workflow-engine:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-isim:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-inventory-mgr:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-core:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-agent:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-telemetry-exporter:10.4.0.0.40620
INFO: docker push 172.16.26.105/repo/atom-scheduler:10.4.0.0.40620
INFO: docker image prune -af
INFO: tar -xf images/vault.tgz
INFO: docker load -qi images/vault/vault-k8s:0.4.0.tar
INFO: docker load -qi images/vault/vault:1.4.2.tar
INFO: docker load -qi images/vault/vault-kubernetes-authenticator:latest.tar
INFO: docker load -qi images/vault/vault-kubernetes-synchronizer:latest.tar
INFO: docker tag hashicorp/vault-k8s:0.4.0 172.16.26.105/repo/vault-k8s:0.4.0
INFO: docker tag vault:1.4.2 172.16.26.105/repo/vault:1.4.2
INFO: docker tag postfinance/vault-kubernetes-authenticator:latest
172.16.26.105/repo/vault-kubernetes-authenticator:latest
INFO: docker tag postfinance/vault-kubernetes-synchronizer:latest
172.16.26.105/repo/vault-kubernetes-synchronizer:latest

```

```
INFO: docker push 172.16.26.105/repo/vault-k8s:0.4.0
INFO: docker push 172.16.26.105/repo/vault:1.4.2
INFO: docker push 172.16.26.105/repo/vault-kubernetes-authenticator:latest
INFO: docker push 172.16.26.105/repo/vault-kubernetes-synchronizer:latest
INFO: docker image prune -af
INFO: tar -xf images/calico.tgz
INFO: docker load -qi images/calico/node:v3.18.1.tar
INFO: docker load -qi images/calico/cni:v3.18.1.tar
INFO: docker load -qi images/calico/pod2daemon-flexvol:v3.18.1.tar
INFO: docker load -qi images/calico/kube-controllers:v3.18.1.tar
INFO: docker tag docker.io/calico/node:v3.18.1
172.16.26.105/repo/node:v3.18.1
INFO: docker tag docker.io/calico/cni:v3.18.1 172.16.26.105/repo/cni:v3.18.1
INFO: docker tag docker.io/calico/pod2daemon-flexvol:v3.18.1
172.16.26.105/repo/pod2daemon-flexvol:v3.18.1
INFO: docker tag docker.io/calico/kube-controllers:v3.18.1
172.16.26.105/repo/kube-controllers:v3.18.1
INFO: docker push 172.16.26.105/repo/node:v3.18.1
INFO: docker push 172.16.26.105/repo/cni:v3.18.1
INFO: docker push 172.16.26.105/repo/pod2daemon-flexvol:v3.18.1
INFO: docker push 172.16.26.105/repo/kube-controllers:v3.18.1
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.**

```

!!!!!!!!!!!!!!!!!!!!!!
Current Progress : 1/5
!!!!!!!!!!!!!!!!!!!!!!
Starting K8s Deployment
Do you want Minimal or Resilient setup?(M/R)M

Enter the Master IP [Info :Virtual IP in case of Resilient setup]:172.16.26.101
wrapper properties updated
Master-0
Enter the IP:172.16.26.101
Worker-0
Enter the IP:172.16.26.102
Worker-1
Enter the IP:172.16.26.103
Worker-2
Enter the IP:172.16.26.104
Master Node list is:
['172.16.26.101']
Worker Node list is :
['172.16.26.102', '172.16.26.103', '172.16.26.104']
Build number is: 10.6.0.0.40929

Updating wrapper.properties.
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
wrapper properties updated
Is 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

1. 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 v3.3.4
 - 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 v1.19
 - i. Installation procedure:
<https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-using-native-package-management>
 - d. Docker installed v18.03 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==7.0.0
2. A site-to-site VPN setup between your datacenter and GCP created for ATOM to reach devices.

3. If the linux box 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.

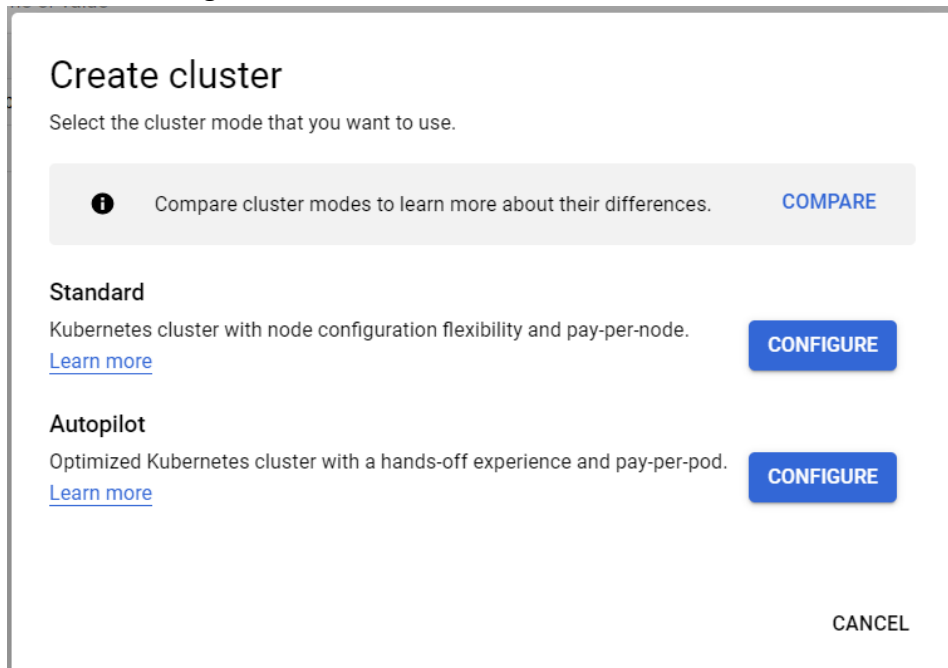
Deploying New K8s Cluster

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

Basic Non-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



- ☐ 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 ?

Location type

- ☒ Zonal
☐ Regional

Zone ▼ ?

- ☐ 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 v1.19

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 ▼

Version ▼

These versions have passed internal validation and are considered production-quality, but don't have enough historical data to guarantee their stability. Known issues generally have known workarounds. [Release notes](#)

- ❑ 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

GENERAL-PURPOSE

COMPUTE-OPTIMIZED

MEMORY-OPTIMIZED

GPU

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

4

Memory

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-tsdbs=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-tsdbs	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.

Basic 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

Standard
Kubernetes cluster with node configuration flexibility and pay-per-node.
[Learn more](#) **CONFIGURE**

Autopilot
Optimized Kubernetes cluster with a hands-off experience and pay-per-pod.
[Learn more](#) **CONFIGURE**

CANCEL

- ❏ 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 v1.19

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)

Version

1.19.9-gke.1400 (default)

These versions have passed internal validation and are considered production-quality, but don't have enough historical data to guarantee their stability. Known issues generally have known workarounds. [Release notes](#)

- 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

GENERAL-PURPOSE

COMPUTE-OPTIMIZED

MEMORY-OPTIMIZED

GPU

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

4

Memory

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.

Prerequisites

4. An Ubuntu machine running with v18.04 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
 - i. Installation procedure: <https://helm.sh/docs/intro/install/>
 - b. AWS CLI v2
 - i. Installation procedure: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install>
 - c. EKSCTL 0.19.0-rc.0
 - i. Installation procedure: <https://github.com/weaveworks/eksctl#installation>
 - d. Kubectl installed v1.16 and above
 - i. Installation procedure: <https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-using-native-package-management>
 - e. Docker installed v18.03 and above
 - i. Installation procedure: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>
 - 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`”

5. A site-to-site VPN setup between your datacenter and AWS VPC created. Refer [Appendix](#) if required.

Login to the Ubuntu machine and follow below steps

- 1) Update the following config in `~/.aws/config` file.
`AWS Access Key ID = xxxxxxxxxxxx`
`AWS Secret Access Key = xxxxxxxxxxxxxxxx`
`Default region name = <us-west-2>`
`Default output format = json`
`role_arn = arn:aws:iam::<AWS-ACCOUNT-ID>:role/<ROLE_NAME>`

[profile default]

AWS Access Key ID and Secret Access Key can be obtained from the IAM/Users section of EKS portal. For first time users, it allows you to download csv.

- 2) Set the environment variables into `/etc/environment` file
KUBECONFIG="/home/<user name in machine>/.kube/config"
AWS_ACCESS_KEY_ID=XXXX
AWS_SECRET_ACCESS_KEY=XXXX

Deploying New K8s Cluster

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

- ❑ Login to your Ubuntu machine and execute below command which does Kubernetes cluster addition with preferred node-type as *r5.xlarge*

```
eksctl create cluster --name {cluster-name} --version 1.19 --region <us-west-2>  
--node-type r5.xlarge --without-nodegroup --node-volume-size 150  
--node-private-networking --vpc-cidr <192.168.64.0/21> --ssh-access=true  
--ssh-public-key={access-key}
```

Ex: `eksctl create cluster --name atomaws --version 1.19 --region us-west-2 --node-type r5.xlarge --without-nodegroup --node-volume-size 150 --node-private-networking --vpc-cidr 192.168.64.0/21 --ssh-access=true --ssh-public-key=aws`

NOTE: If ssh public key is missing then generate using `ssh-keygen`
If any resource issue happens use “`--zones=us-west-2a,us-west-2b`”

- ❑ Delete the Amazon VPC CNI
`kubectrl 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.

```
kubectrl apply -f calico.yaml
```

- ❑ Create EC2 instances
Based on type of setup being created, execute respective command

Basic Non-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 150 --cluster
{cluster-name} --node-private-networking --node-type r5.xlarge --managed
```

Example:

```
eksctl create nodegroup --name awsatomgroup --nodes 3 --nodes-min 3 --nodes-max 3
--node-volume-size 150 --cluster atomaws --node-private-networking --node-type
r5.xlarge --managed
```

Basic 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 150 --cluster
{cluster-name} --node-private-networking
--node-labels=infra-tsdb-2=deploy,object_store=deploy,monitoring_server=deploy
--node-zones {availability-zone-name} --node-type r5.xlarge --managed
```

- For resilient setup labels for nodes will be assigned during creation of the nodegroup itself.
- For resilient setup, cluster creation happens by default across 3 availability zones (AZ) and 2 nodegroups will be created in 3 different AZ, effectively every nodegroup contains one VM because of labelling.

Example:

```
eksctl create nodegroup --name awsatomgroup1 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=broker=deploy,elasticsearch=deploy,zookeeper=deploy,object_store=deploy
--node-zones us-west-2a --node-type r5.xlarge

eksctl create nodegroup --name awsatomgroup2 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=distributed_db=deploy,default_agent=deploy,object_store=deploy,grafana=
deploy --node-zones us-west-2a --node-type r5.xlarge

eksctl create nodegroup --name awsatomgroup3 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=distributed_db=deploy,infra-tsdb-1=deploy,agent1=deploy,securestore=dep
loy --node-zones us-west-2b --node-type r5.xlarge
```

```
eksctl create nodegroup --name awsatomgroup4 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=broker=deploy,zookeeper=deploy,elasticsearch=deploy,northbound=deploy,object_store=deploy --node-zones us-west-2b --node-type r5.xlarge

eksctl create nodegroup --name awsatomgroup5 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=broker=deploy,zookeeper=deploy,elasticsearch=deploy,northbound=deploy,object_store=deploy --node-zones us-west-2d --node-type r5.xlarge

eksctl create nodegroup --name awsatomgroup6 --nodes 1 --nodes-min 1 --nodes-max 1
--node-volume-size 150 --cluster atomaws --node-private-networking
--node-labels=infra-tsdb-2=deploy,monitoring_server=deploy,object_store=deploy
--node-zones us-west-2d --node-type r5.xlarge
```

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

❑ 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.

❑ VPC association with Virtual private gateway

Services -> Networking & Content Delivery -> VPC -> Virtual Private Network -> Virtual Private Gateway

Right click on entry (xxx-vgw) and Attach to VPC (select the VPC which is created as part of step1)

Note: No need to create any new entry and don't delete existing entry xxx-vgw

❑ Allow Access to Remote-ATOM-Agents from the AWS ATOM Server

- a. From the AWS console select **Services > Networking & Content Delivery > VPC** > under **Security** section: select **Security Groups** > Create a required security group by clicking **Create Security Group**.

Refer to [Networking requirements](#) section Inter-components communication table.

- b. From the AWS console select **Resources > Compute > EC2** > under **INSTANCES** section: select **Instances** > select instances which we need to enable ICMP and SSH access or any other services > Select **Actions** > Select **Networking** > select **Change Security Groups** and assign the security group to the instance

8. Route Propagation

From AWS console select **Resources > Networking & Content Delivery > VPC >** under **Virtual Private Cloud** section: select **Route Tables** > select all the entries which are associated with above created VPC(mainly <3/6> private entries, one per region) > Select **Route Propagation** > Click on **Edit Route Propagation** and enable Propagate option.

Deploying ATOM

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

5. 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}”
```

- b. Get all running node labels by executing

```
“kubectl get nodes --show-labels”
```

- c. Add labels using

```
“kubectl label node <node name> <key 1>=deploy <key 2>=deploy”
```

where keys are labels such as broker, northbound, infra-distributed-db, infra-elasticsearch and so on.

For Minimal ATOM setup

```
kubectl label node <node name 1> northbound=deploy infra-tsdb-1=deploy
monitoring_server=deploy elasticsearch=deploy
```

```
kubectl label node <node name 2> default_agent=deploy object_store=deploy
```

```
kubectl label node <node name 3> broker=deploy zookeeper=deploy
distributed_db=deploy grafana=deploy securestore=deploy
```

```
Ex: kubectl label node ip-192-168-67-146.us-west-2.compute.internal
default_agent=deploy object_store=deploy
```

For Resilient ATOM setup

```
NA. As already done in nodegroup creation
```

6. 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 **“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 permissions to log file by running. *"sudo touch /var/log/atom.log && sudo chmod 777 /var/log/atom.log"*
7. Run the K8s secrets using below command

```
# kubectl apply -f scripts/vault/secrets
```
8. Deploy ATOM by executing the following script

```
# sudo python deploy_atom.py
```
9. Goto scripts folder and execute below to setup SSO

```
# sh deploySSO.sh
```
5. 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
```

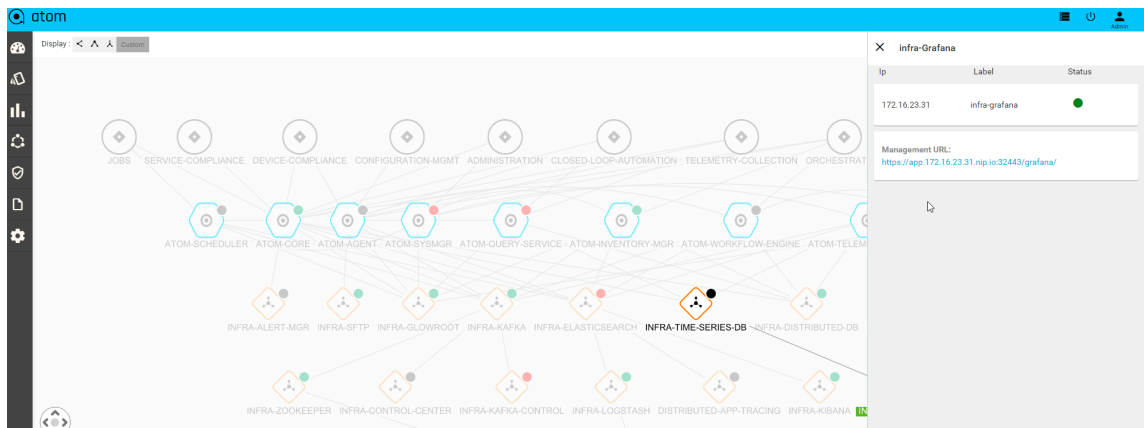
For local deployments, the services are accessible via nodePorts and for cloud deployments the services are accessible via LoadBalancers.

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.

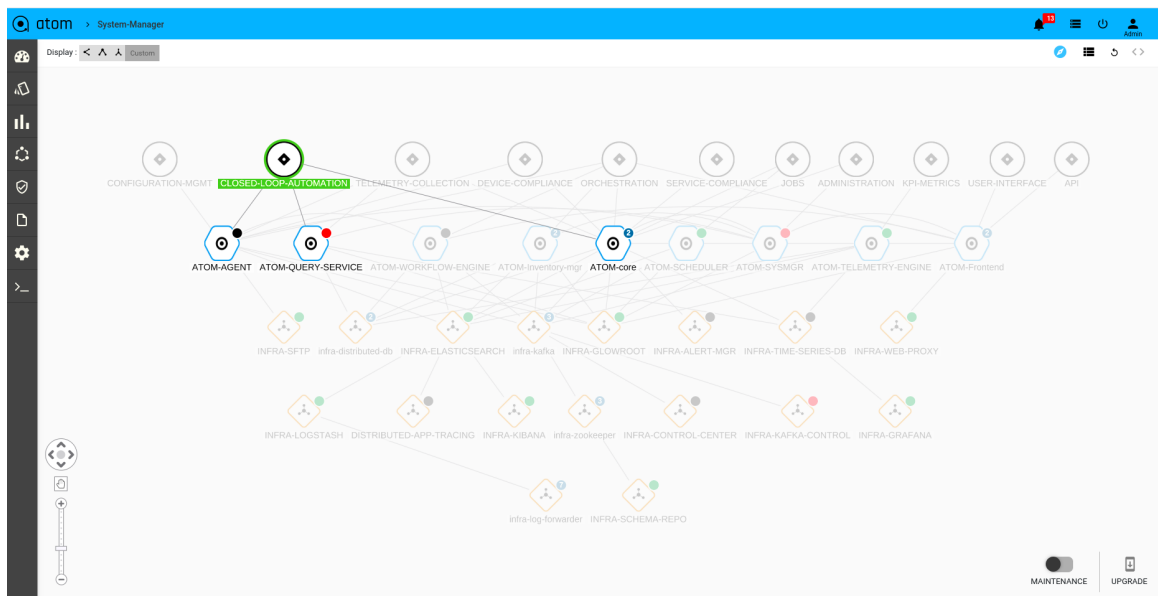
ATOM System Manager

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.

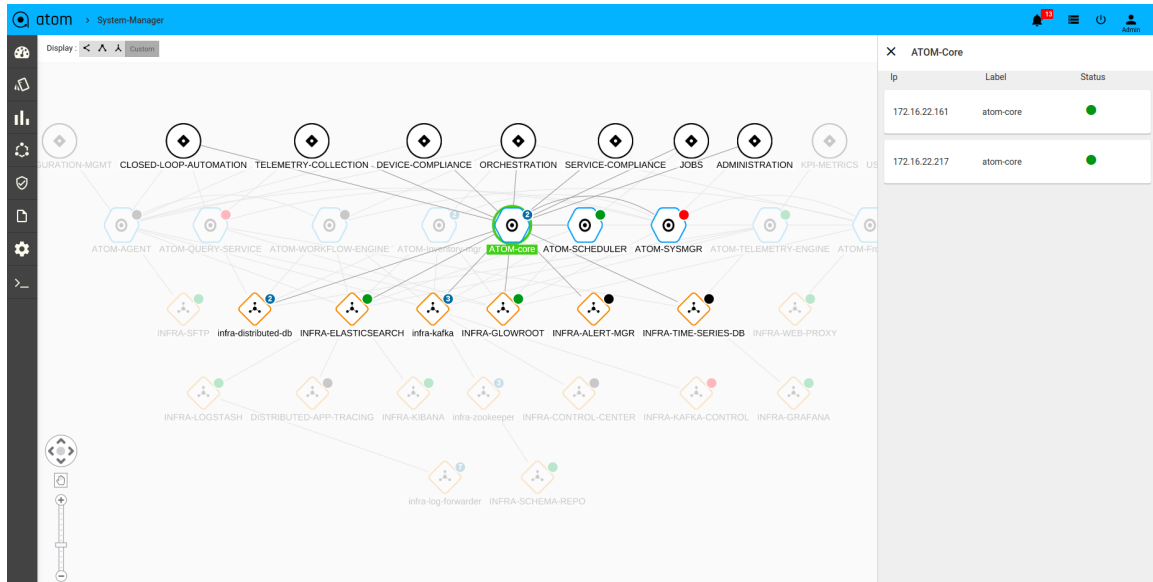
- Navigate to Administration> System Manager> Dashboard
- To access components like Grafana, Kibana etc. select that component and you can see the Management URL at top right corner.



- 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.



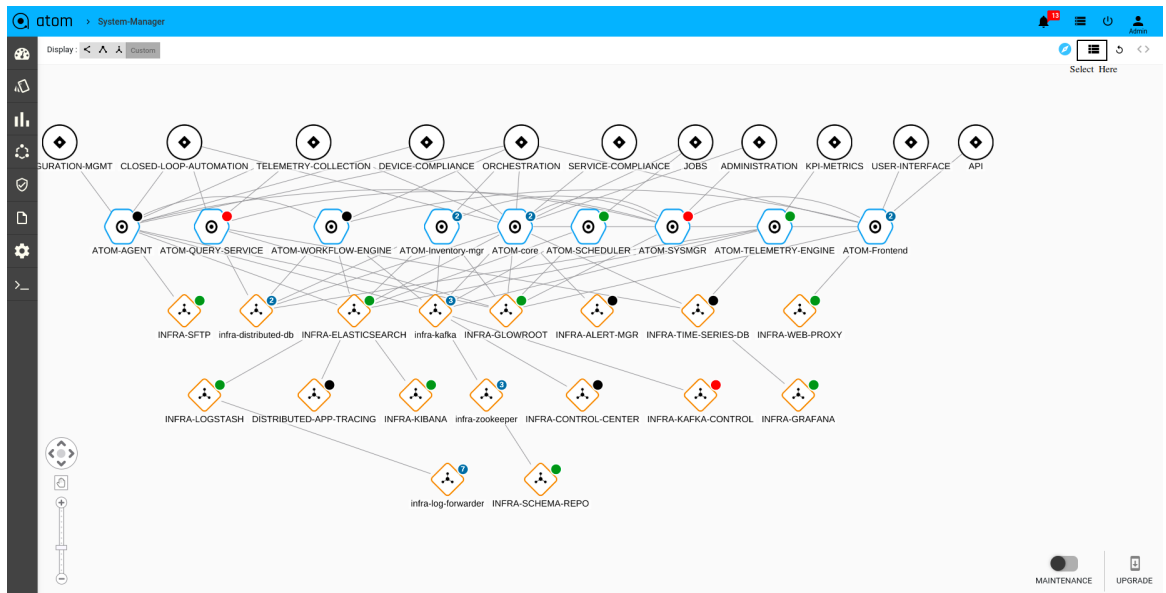
Here on the right side we can see the number instances, refer below colour codings.

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



- 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-ism	0	Down	
atom-prp-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-gvc	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-tadb-1-alertmanager	1	Ready	
infra-tadb-monitoring-alertmanager	1	Ready	
infra-tadb-monitoring-kube-state-metrics	1	Ready	
infra-tadb-query	2	Ready	

Post Installation

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	<a href="https://<master-ip>:32443/k8s/">https://<master-ip>:32443/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	<a href="https://<master-ip>:30443">https://<master-ip>:30443 (for local users) <a href="https://<master-ip>:32443/">https://<master-ip>:32443/ (for SSO users) <a href="https://<FQDN>">https://<FQDN> (for cloud users)
Grafana	Service which helps us in monitoring infrastructure health using heapster and time series database.	<a href="https://<master-ip>:32443/grafana/">https://<master-ip>:32443/grafana/ (for SSO users) <a href="https://<FQDN>/grafana">https://<FQDN>/grafana (for cloud users)
Kibana	Service which helps us in log monitoring and analytics	<a href="https://<master-ip>:32443/kibana/">https://<master-ip>:32443/kibana/ (for SSO users) <a href="https://<FQDN>/kibana">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/admin can be used. Additional users can be created by login to the atom authentication manager ui
<https://<master-ip>:32443/auth> and the default credentials are admin/admin
- 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.

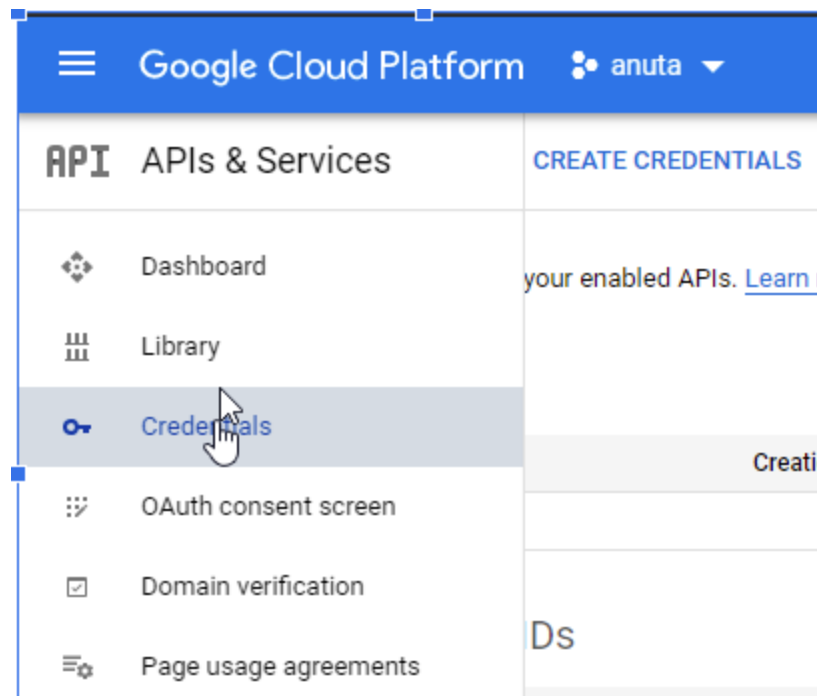
Default `<master_ip>` is configured during ATOM setup with self signed certificates. If a specific domain is desired, then users can provide a FQDN address for the master IP of K8s cluster and an SSL certificate associated with the same FQDN in PEM format.

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

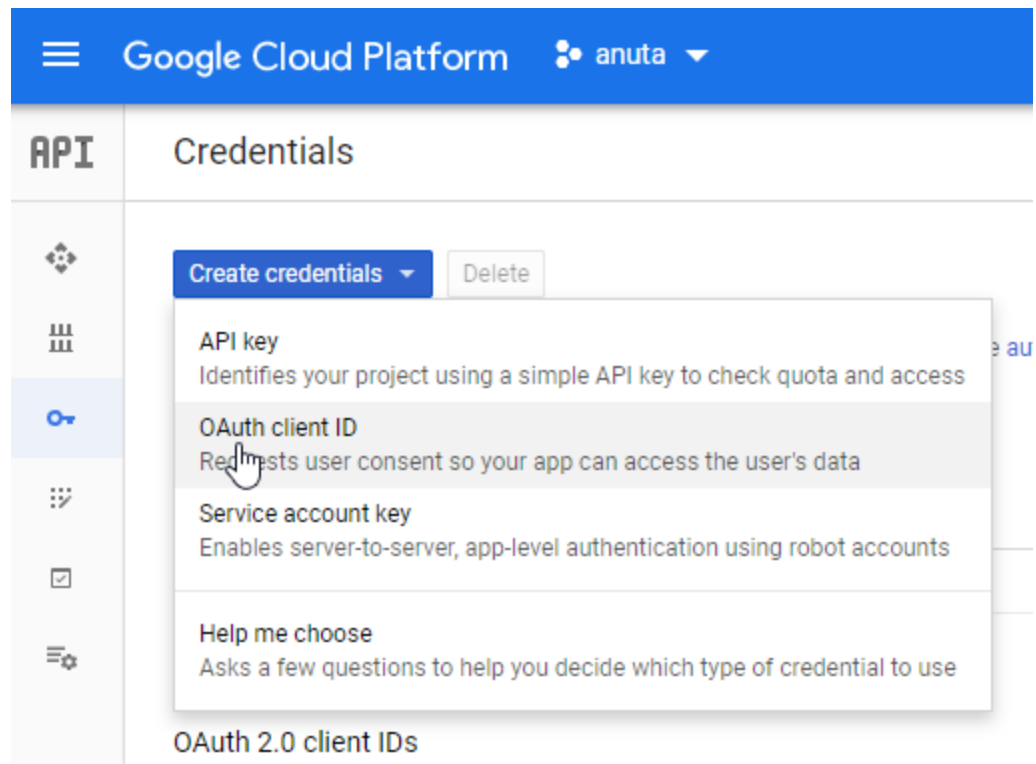
Google IdP

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

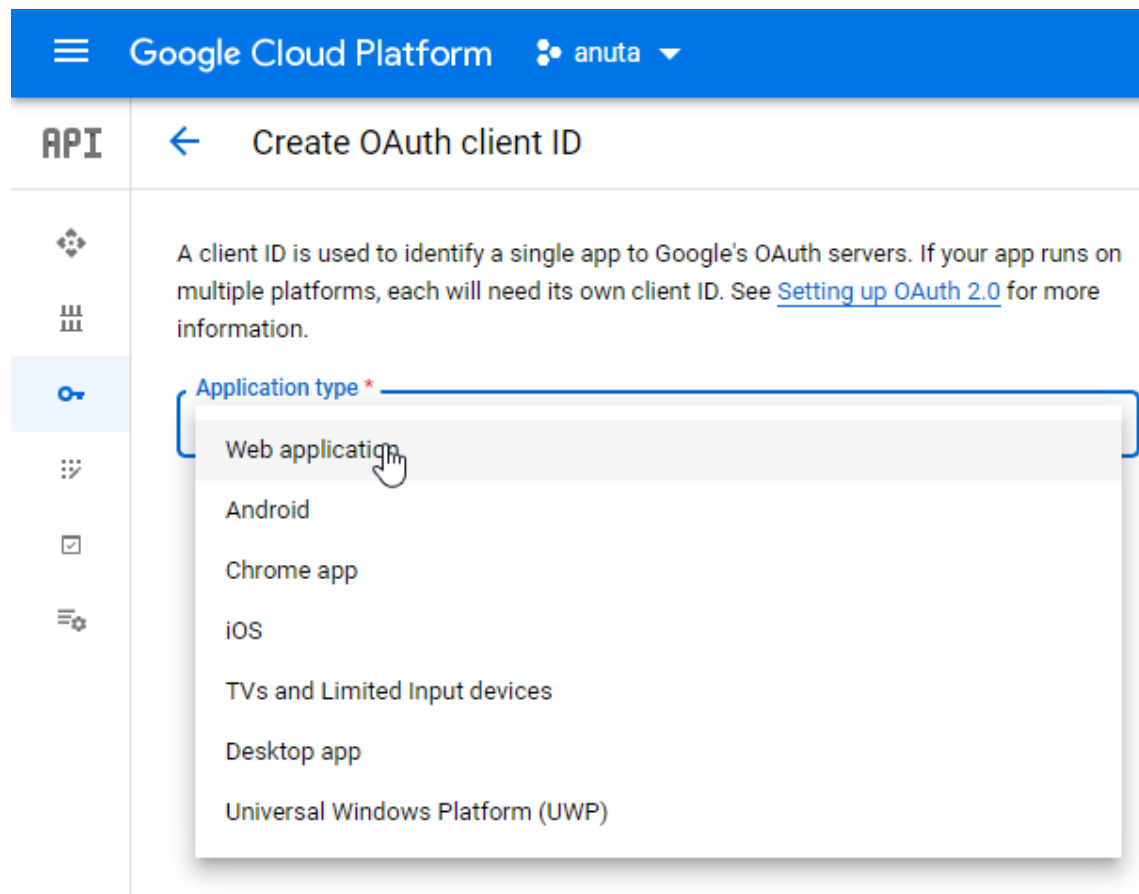
- Login to Google cloud platform. In the project Dashboard center pane, choose APIs & Services tab.
- In the left navigation pane, choose “Credentials”



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



- Select application type as web application



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

Google Cloud Platform

anuta

API

←

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

+ ADD URI

Authorized redirect URIs ?

For use with requests from a web server

URIs

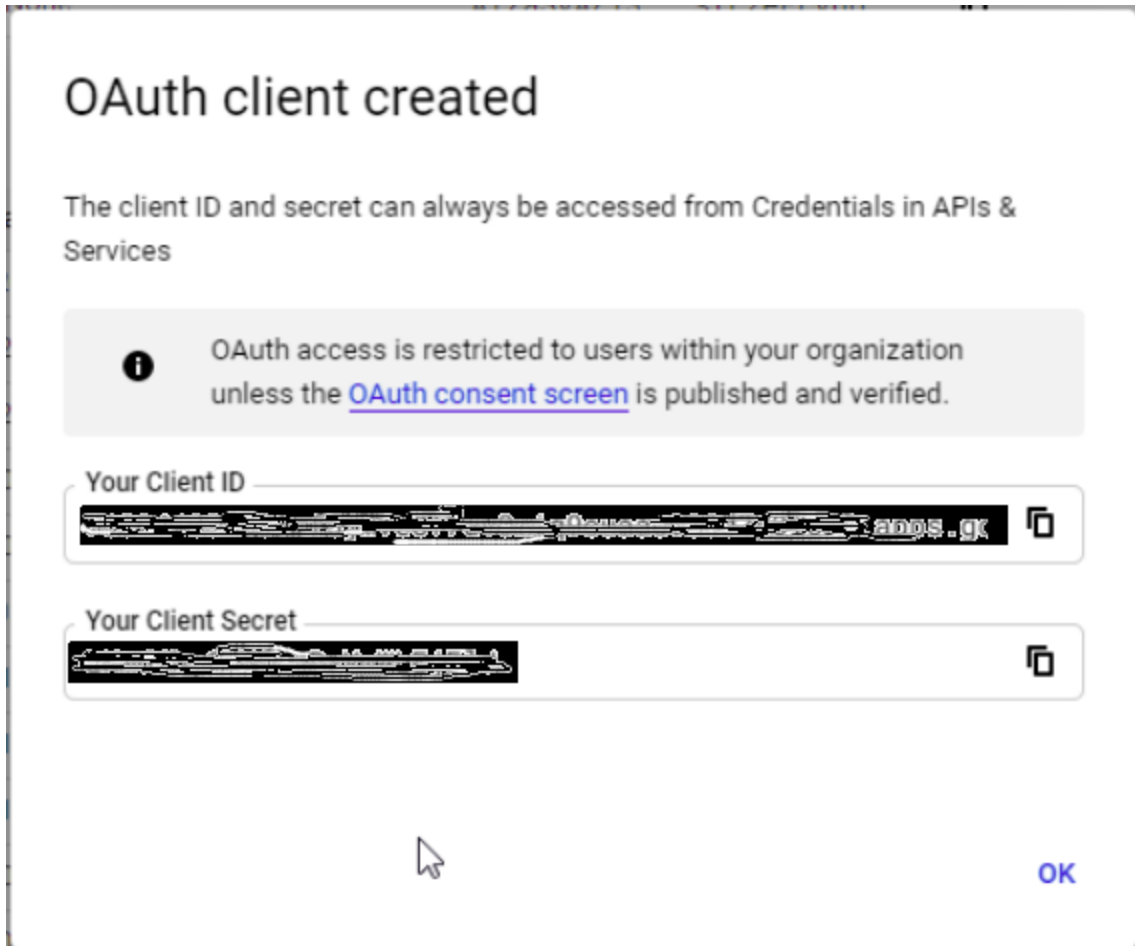
https://app.172.16.22.16.nip.io:32100/callback

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

+ ADD URI

CREATE

CANCEL



- 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	<ul style="list-style-type: none"> • Login to Grafana (<a href="https://<Master IP>:32443/grafana/">https://<Master IP>:32443/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	<ul style="list-style-type: none"> • Login to Grafana (<a href="https://<Master IP>:32443/grafana/">https://<Master IP>:32443/grafana/) • Select Cluster Health dashboard

	<ul style="list-style-type: none"> ● Select the node which has HighCPU alert and check which are the pods consuming more CPU.
NodeHighDiskUsage	<ul style="list-style-type: none"> ● Login to Grafana. ● Select Cluster Monitoring Dashboard. ● Select the node which has High Disk usage and login to that node. <ul style="list-style-type: none"> ○ Login to Master VM ○ Find the IP of the node by following command. <ul style="list-style-type: none"> ■ 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 <ul style="list-style-type: none"> ○ df -h which will give full disk utilization. (/dev/mapper/centos-root is the filesystem we need to check)
NodeFault	<p>This alert can be received for the following reasons.</p> <ul style="list-style-type: none"> ● 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. <p>If you receive this Alert follow the steps below.</p> <ul style="list-style-type: none"> ● 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	<p>This alert will come when the Node/VM is not reachable or down. If you receive this Alert, intimate to Admin.</p>
PodNotReady	<p>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</p> <p>Following can be the reasons for PodNotReady</p>

	<ul style="list-style-type: none"> ● 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	<p>Below can be possible reasons for Container not being up.</p> <ul style="list-style-type: none"> ● Crashloopbackoff ● ImagePullbackoff ● Application inside the container was not up. <p>When this Alert is generated follow the steps below.</p> <ul style="list-style-type: none"> ● Note the reason shown for the Container not up and pod-name ● Login to the Kibana (<a href="https://<Master IP>:32443/kibana/">https://<Master IP>:32443/kibana/) ● Filter by pod name ● Verify the logs and find if any error messages are shown.
ContainerTerminated	<p>Containers can be killed for a number of reasons like OutofMemory (OOMKilled), Eviction, DiskPressure etc..</p> <p>When this Alert is generated follow the steps below.</p> <ul style="list-style-type: none"> ● Note the reason shown for the Container termination and pod-name ● Login to the Kibana (<a href="https://<Master IP>:32443/kibana/">https://<Master IP>:32443/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	<p>This alert can be received when one or more pods are</p> <ul style="list-style-type: none"> ● Not ready due to crashed application internally ● In pending state due to missing resource ● In failed container state <p>When this Alert is generated follow the steps below</p>

- 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-tdsb-monitoring-alertmanager by following below steps:

- Login to the Master IP through ssh and execute below command.

kubectl edit configmap infra-tdsb-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 }} &#x2014; {{ .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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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

List of useful commands

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

Command	Description
<code>helm create {package_name}</code>	To create a directory
<code>helm install {package_name} -n {name of app}</code>	Deploy the application
<code>helm ls -n atom</code>	To check deployment status
<code>Kubectl get deployments -n atom</code>	To check deployments
<code>Kubectl get pods -n atom</code>	To check pod status

<code>helm upgrade {releasename} {package-name} -n atom</code>	deployment update with new changes.
<code>helm history {package-name} -n atom</code>	To view the history
<code>helm uninstall {package_name} -n atom</code>	To delete the package
<code>helm rollback {package-name} version -n atom</code>	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 and if you are working on a remote KVM box without a GUI tool like vm manager, follow below steps

- Master node

```
virt-install --name "<VM_NAME>" --memory <RAM_IN_MB> --vcpus <CPU_COUNT> --disk
<FULL_PATH_OF_MASTER_QCOW2_IMAGE> --network=<BRIDGE_NAME_AND_TYPE> --vnc
--import
```

ex:

```
virt-install --name "master" --memory 8192 --vcpus 4 --disk
/home/anuta/Downloads/master/centos_1_18_40.qcow2
--network=bridge:virbr0,model=virtio --vnc --import
```

- For each worker node with linstor disks

```
virt-install --name "<VM_NAME>" --memory <RAM_IN_MB> --vcpus <CPU_COUNT> --disk
<DISK1_QCOW2_IMAGE> --disk <DISK2_QCOW2_IMAGE> --network=<BRIDGE_NAME_AND_TYPE>
--vnc --import
```


ex:

```
virt-install --name "worker1" --memory 32768 --vcpus 4 --disk  
/home/anuta/Downloads/worker1/centos_1_18_300_linstor-disk1.qcow2 --disk  
/home/anuta/Downloads/worker1/centos_1_18_300_linstor-disk2.qcow2  
--network=bridge:virbr0,model=virtio --vnc --import
```

In case of Worker Nodes when we have Multiple disks.Do make sure that

`centos_1_18_300_linstor-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_18_300_linstor-disk1.qcow2` which maps to (sd|hd)a, while Data disk always maps to `centos_1_18_300_linstor-disk2.qcow2` which maps to (sd|hd)b.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:

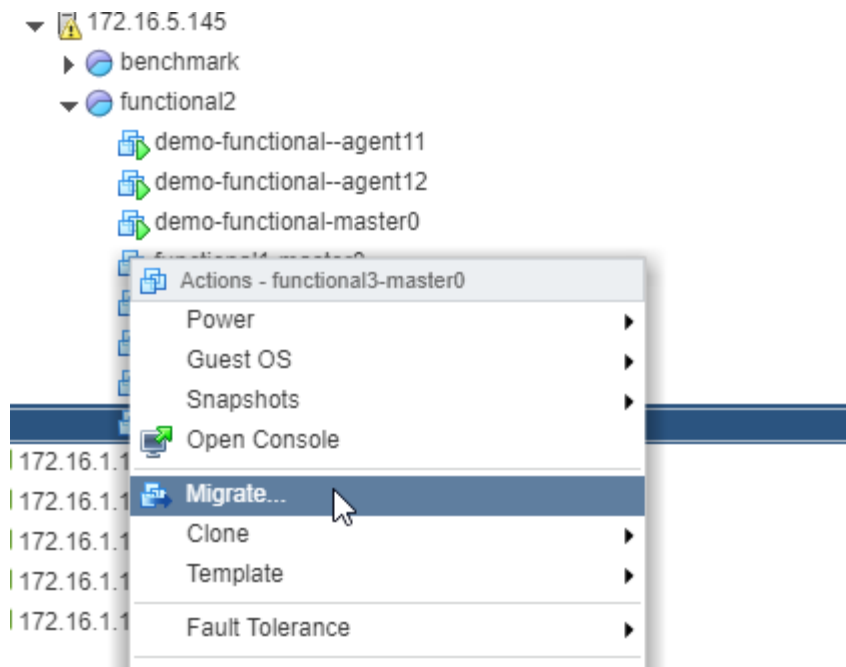
1. Migrate VM and change the Data Storage - for example, nodes running on esxi hosts can be migrated using vSphere.
2. Power on the Nodes

STEP-3 Post VM Migration Steps in ATOM:

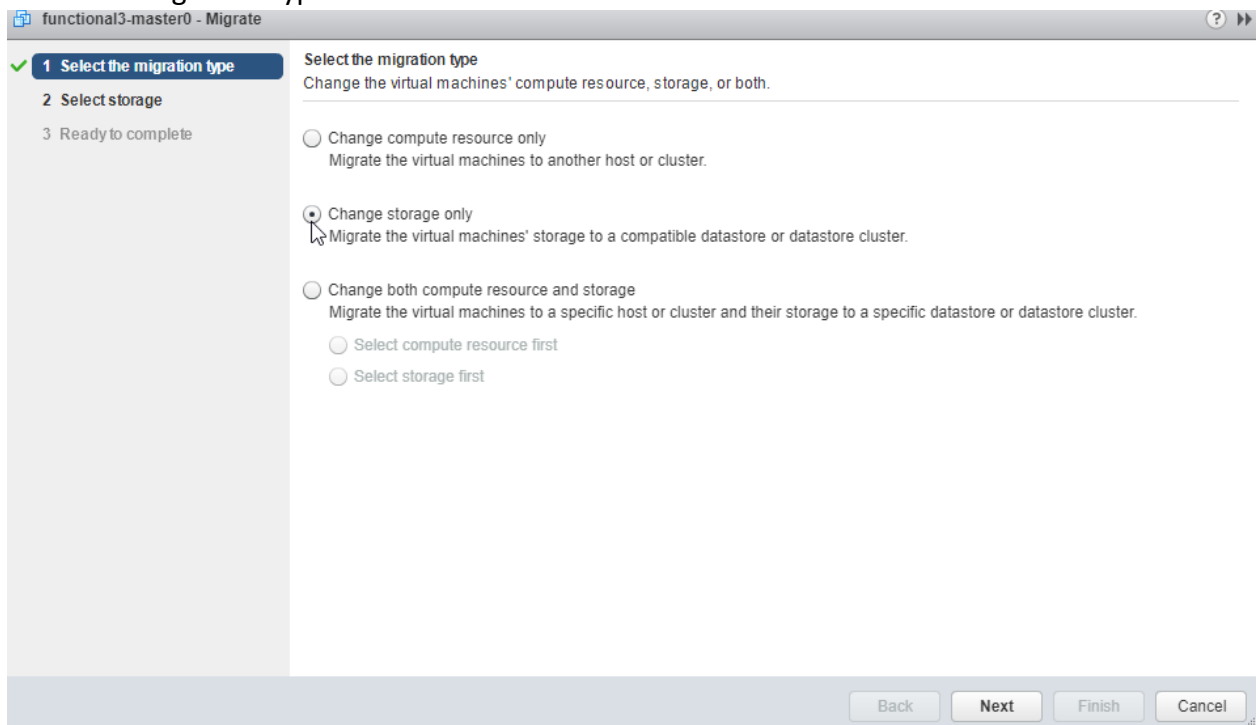
1. Remove the maintenance mode.
Navigate to Administration > System Manager > Dashboard. Disable "Maintenance" option to clear the system from maintenance mode.

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.



3. Select the storage type.

functional3-master0 - Migrate

1 Select the migration type
2 **Select storage**
3 Ready to complete

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	

Advanced >>

Compatibility
 ✓ Compatibility checks succeeded.

Back Next Finish Cancel

- Click on the Next and Finish button.

Steps to check logs in kibana

- Open the kibana url , http://<master_ip>:32443/kibana/
- Create index pattern by going to Management/Index patterns/Create index pattern

Management / Index patterns / Create index pattern

Elasticsearch
 Index Management
 Index Lifecycle Policies
 Rollup Jobs
 Cross-Cluster Replication
 Remote Clusters
 Snapshot and Restore
 License Management
 8.0 Upgrade Assistant

Kibana
 Index Patterns
 Saved Objects
 Spaces
 Reporting
 Advanced Settings

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

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 \, /, *, <, >, |.

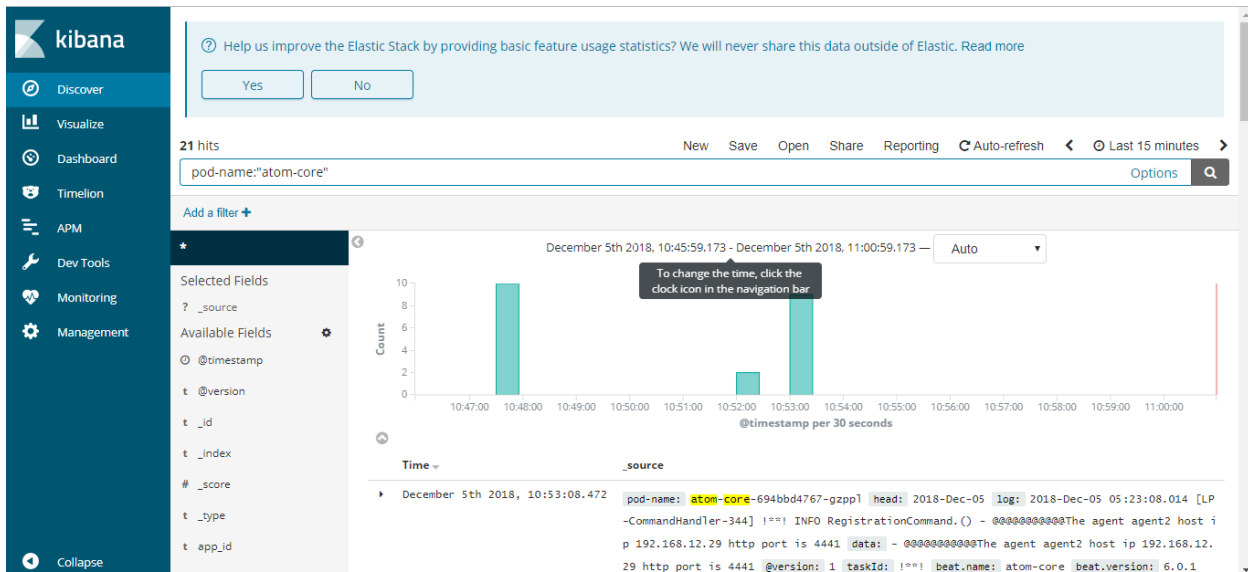
Your index pattern can match any of your 2 indices, below:

- logs
- tasklogs

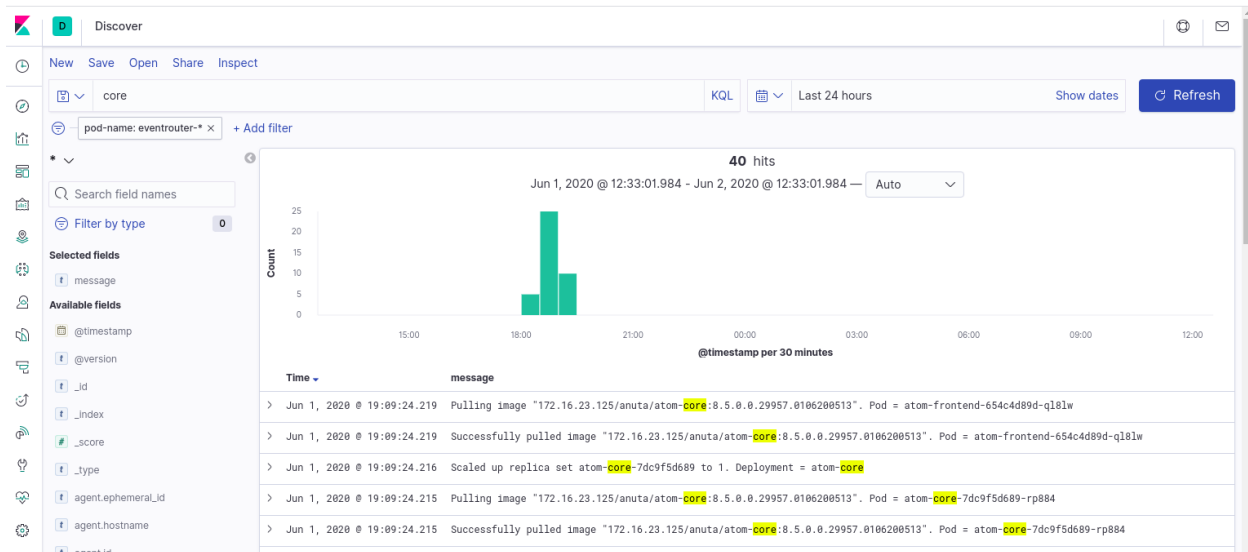
Rows per page: 10

> Next step

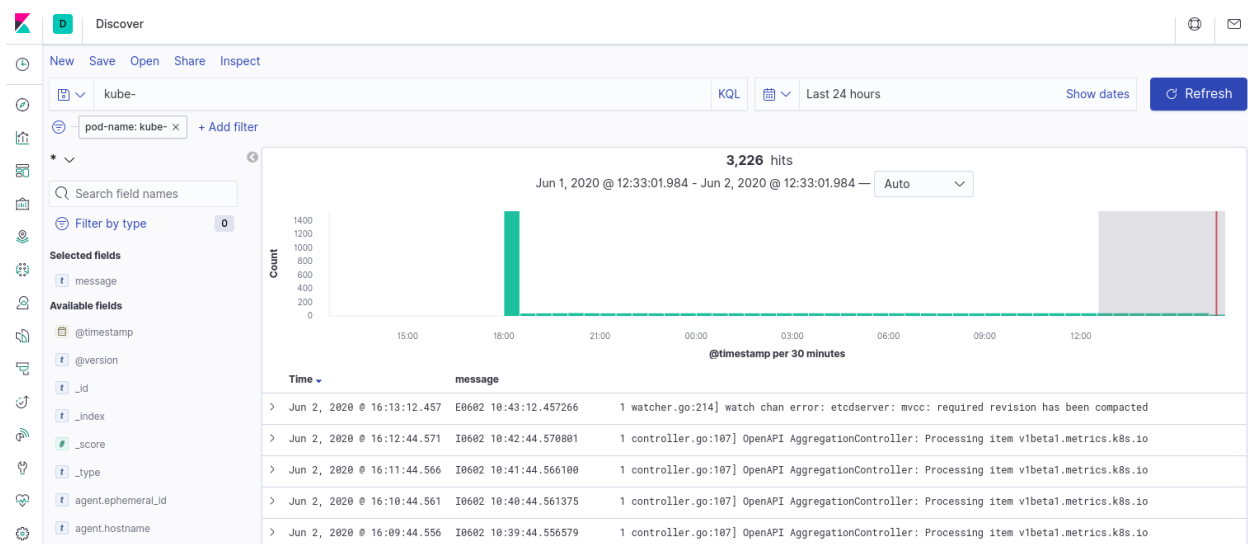
- Go to Discover and in the search box , search with pod name as shown below
 pod-name:"<pod-name>", to check logs for specific pod



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



For K8s pod logs



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

Site-to-Site VPN Setup

To Create a site-to-site vpn connection between Datacenter and the AWS VPC follow the below steps.

Create a VPC network

From AWS console select **Resources > Networking & Content Delivery > VPC > under Virtual Private Cloud** section: select **Your VPCs > Create a required VPC** by clicking **Create VPC**

Create Customer Gateway

From AWS console select **Resources > Networking & Content Delivery > VPC > under Virtual Private Network (VPN)** section: select **Customer Gateways > Create a required gateway** by clicking **Create Customer Gateway**

Create Virtual Private Gateway

1. From AWS console select **Resources > Networking & Content Delivery > VPC >** under **Virtual Private Network (VPN)** section: select **Virtual Private Gateways** > Create a virtual gateway(aws cloud side) by clicking **Create Virtual Private Gateway**
2. Once Created virtual private gateway, attach to a VPC by right clicking on the entry

Create Site-to-Site VPN Connection

From AWS console select **Resources > Networking & Content Delivery > VPC >** under **Virtual Private Network (VPN)** section: select **Site-to-Site VPN Connections** > Create a VPN tunnel between customer gateway and virtual private gateway by clicking **Create VPN Connection**

While creating VPN connections provide customer private networks.

Enable Route Propagation

From AWS console select **Resources > Networking & Content Delivery > VPC >** under **Virtual Private Cloud** section: select **Route Tables** > select an entry which is associated with created VPC > Select **Route Propagation** > Click on **Edit Route Propagation** and enable Propagate option.

Configure Customer Site Gateway

1. Download the configuration corresponding to the customer gateway device by selecting VPN connection which is created at step4.
2. The downloaded configuration needs to be configured on the customer gateway device. As an Example it can be related to configuring IPsec Tunnels, ACLs, IPsec Configuration, Static Routes.

Enable ICMP access to AWS EC2 instances(Optional)

1. From the AWS console select **Resources > Networking & Content Delivery > VPC >** under **Security** section: select **Security Groups** > Create a required security group by clicking **Create Security Group**
2. From the AWS console select **Resources > Compute > EC2 >** under **INSTANCES** section: select **Instances** > select instances which we need to enable ICMP and SSH access >

Select **Actions** > Select **Networking** > select **Change Security Groups** and assign the security group.

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. Connect to the k8s cluster as **atom** user
2. Copy the certificate and private key to the host node

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

3. Delete existing secrets

```
kubectrl delete secret -n atom atom-certificate
kubectrl delete secret -n kube-system tls-secret
```

4. Create new secrets using the new certificate and private key.

```
kubectrl create secret tls atom-certificate -n atom --cert=<certificate-name>
--key=<private-key-name> --dry-run -o yaml >cert-atom.yaml
kubectrl create secret tls tls-secret -n kube-system --cert=<certificate-name>
--key=<private-key-name> --dry-run -o yaml >cert-dashboard.yaml
```

NOTE: Replace <certificate-name> and <private-key-name> with your files.

5. Apply the secrets

```
kubectrl create -f cert-atom.yaml -f cert-dashboard.yaml
```