

AWS KUBO Deploying Guide

Abhilash S

(주)크로스েন্ট
2018. 06

Agenda

1. AWS Infrastructure Setting
2. Deploying Bosh Director on AWS
3. Deploying KUBO on AWS
4. KUBO Deployment Architecture on AWS

Note: This deployment Guide is targeted only Ubuntu 16.04 users.

1. AWS Infrastructure Setting (1/12)

- Creating VPC (1/2)

Note: Install and Configure below Steps on your AWS Console.

❖ Create Elastic IPS in your Preferred Zone.

* AWS Services → EC2 → Elastic IPs → Allocate new address → Allocate

❖ Create VPC Wizard in your Preferred Zone.

Step 1: Select a VPC Configuration

VPC with a Single Public Subnet

1 VPC with Public and Private Subnets

VPC with Public and Private Subnets and Hardware VPN Access

VPC with a Private Subnet Only and Hardware VPN Access

In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT).

Creates:

A /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via Network Address Translation (NAT). (Hourly charges for NAT devices apply.)

2 Select

Internet, S3, DynamoDB, SNS, SQS, etc.

Amazon Virtual Private Cloud

Public Subnet

Private Subnet

NAT

* Select VPC with Public and Private Subnets

1. AWS Infrastructure Setting (2/12)

- Creating VPC (2/2)

Step 2: VPC with Public and Private Subnets

IPv4 CIDR block* 10.0.0.0/16 2531 IP addresses available

IPv6 CIDR block: ☐ Use IPv6 CIDR block ☐ Amazon provided IPv6 CIDR block

VPC name: Kubo VPC

Public subnet's IPv4 CIDR* 10.0.0.0/24 251 IP addresses available

Availability Zone* ap-northeast-2a 1

Public subnet name: Kubo Public subnet

Private subnet's IPv4 CIDR* 10.0.1.0/24 251 IP addresses available

Availability Zone* ap-northeast-2a 1

Private subnet name: Kubo Private subnet

You can add more subnets after AWS creates the VPC.

Specify the details of your NAT gateway (NAT gateway name optional)

Elastic IP Allocation ID* eipalloc-ed96adc3

Service endpoints

Enable DNS hostnames* ☒ Yes ☐ No

Hardware strategy* ☐ Classic ☒ VPC

* CIDR range should be /16 netmask

- Public Subnet CIDR range should be /24 netmask
- Select Your Availability Zone
- Name your Public Subnet
- Private Subnet CIDR range should be /24 netmask
- Select Your Availability Zone
- Name your Private Subnet

* Select Elastic IP, that created in previous steps

* Make Sure to enable DNS hostnames

Cancel and Exit Back Create VPC

Note: It will take 2 ~ 3 Minutes to create VPC Wizard.

❖ Check your VPC is created.

* AWS Services → VPC → Your VPCs

1. AWS Infrastructure Setting (3/12)

- Configure Tags for Subnets (1/2)

❖ Move to Subnets TAB.

* AWS Services → VPC → Subnets

❖ Create Tags for Your Private Subnets.

* AWS Services → VPC → Subnets → Kubo Private subnet → Tags → Edit

The screenshot shows the AWS Management Console interface for configuring tags on a subnet. At the top, a search bar and a list of subnets are visible. The 'Kubo Private subnet' is selected. Below the list, the 'Tags' tab is active. A table shows the existing tag 'Name' with the value 'Kubo Private subnet'. A new tag 'KubernetesCluster' with the value 'kubocnexpert' is being added. A red dashed box highlights the new tag entry, and a red arrow points to it from annotation 3. Annotation 1 points to the subnet selection in the list, and annotation 2 points to the 'Tags' tab.

Key	Value	Remove
Name	Kubo Private subnet	
KubernetesCluster	kubocnexpert	

[Add another Tag](#) (Maximum of 9)

- This Tag is used in Kubo Deployment
- Make Sure to have same Tag Key and Value for Private and Public Subnets

1. AWS Infrastructure Setting (4/12)

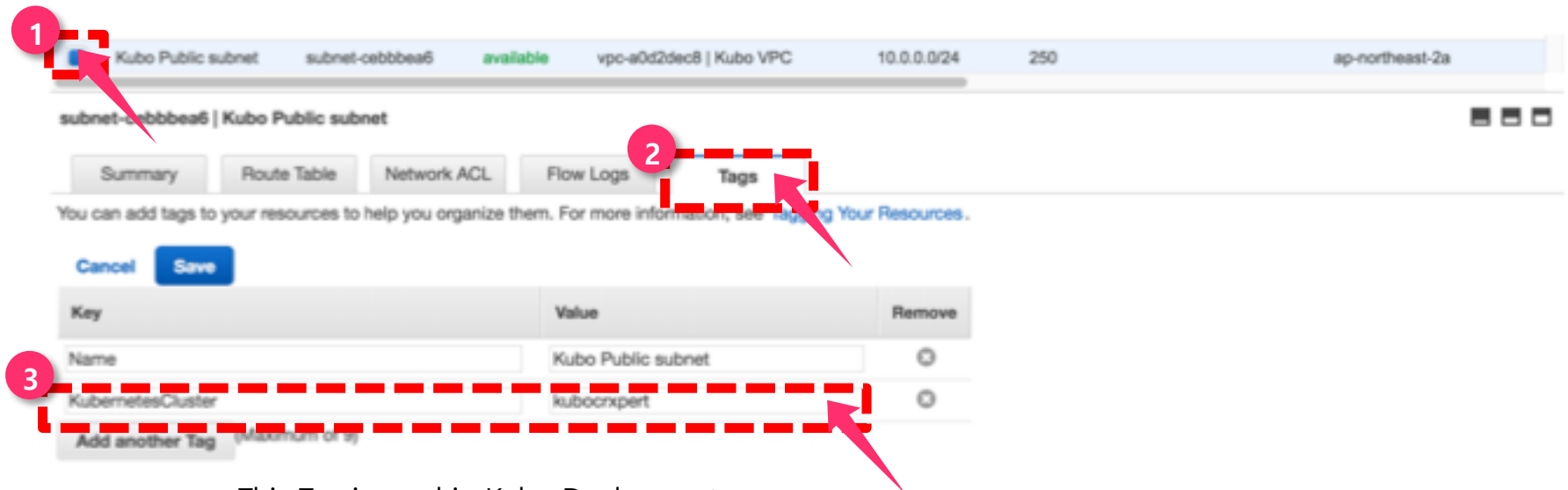
- Configure Tags for Subnets (2/2)

❖ Move to Subnets TAB.

* AWS Services → VPC → Subnets

❖ Create Tags for Your Private Subnets.

* AWS Services → VPC → Subnets → Kubo Public subnet → Tags → Edit



- This Tag is used in Kubo Deployment
- Make Sure to have same Tag Key and Value for Public and Private Subnets

1. AWS Infrastructure Setting (5/12)

- Configure Route Tables

❖ Move to Route Tables TAB.

* AWS Services → VPC → Route Tables

❖ Create Route Table for Your Private Subnet.

* AWS Services → VPC → Route Tables

❖ Create Routes in your Created Route Table.

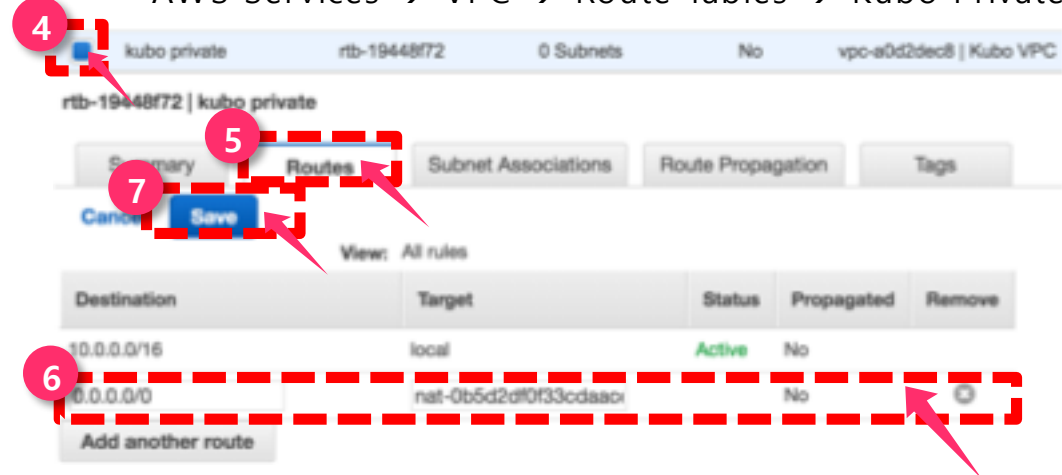
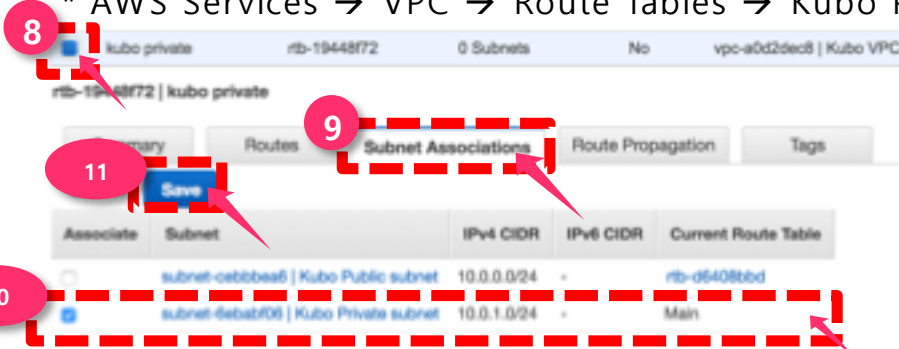
* AWS Services → VPC → Route Tables → Kubo Private



- Name Your Route Table
- Select your VPC created in previous steps

❖ Edit Subnet Associations in your Route Table.

* AWS Services → VPC → Route Tables → Kubo Private



- Make Sure to select Nat Gateway where our deployment access to internet through Nat Gateway.

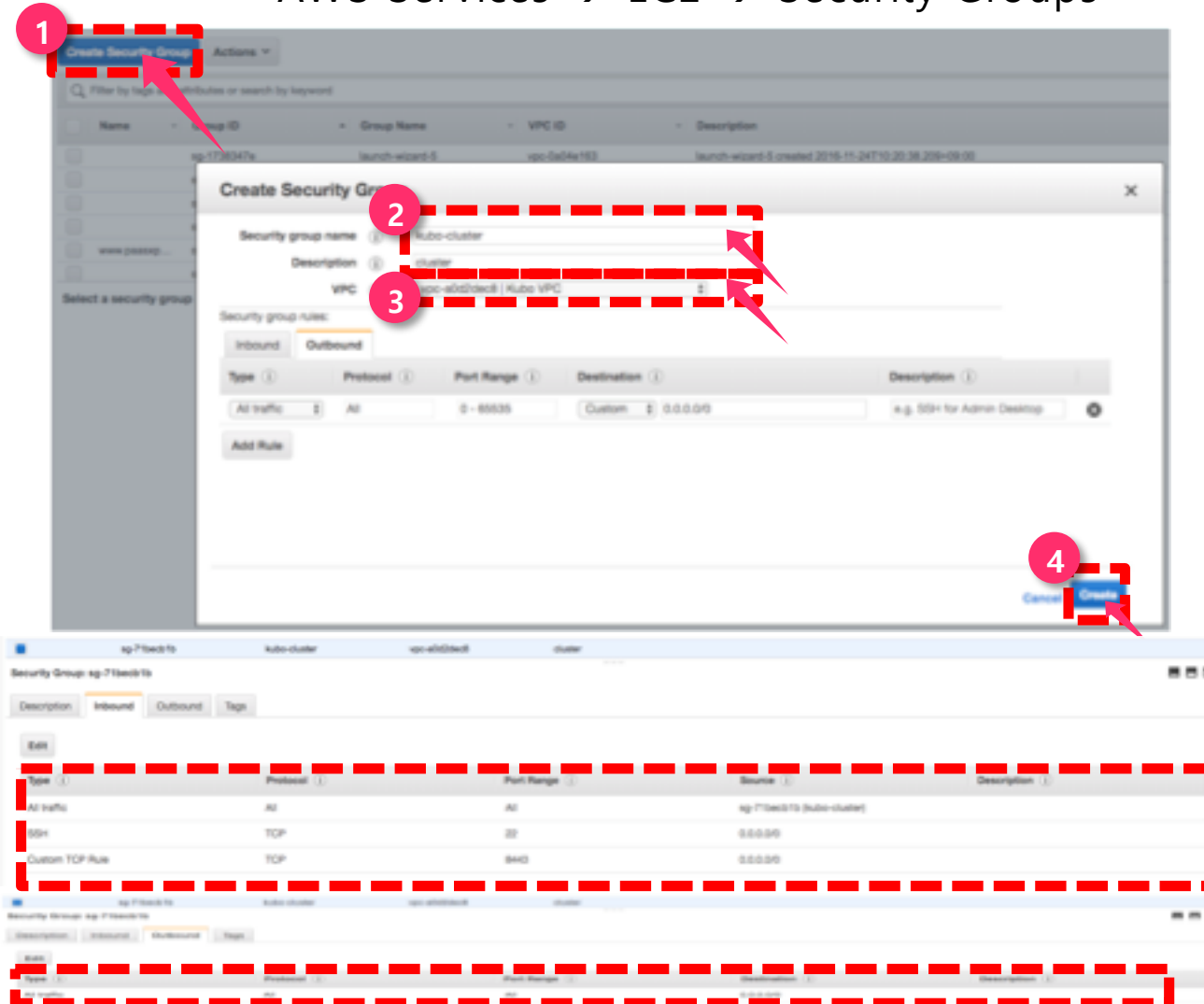
- Select Your Private Subnet where we deploy Micro-Bosh, and Kubo.

1. AWS Infrastructure Setting (6/12)

- Create Security Groups (1/2)

❖ Move to Security Group TAB.

* AWS Services → EC2 → Security Groups



- This Security Group is Used only for Inception, Micro-Bosh, Kube VM's.

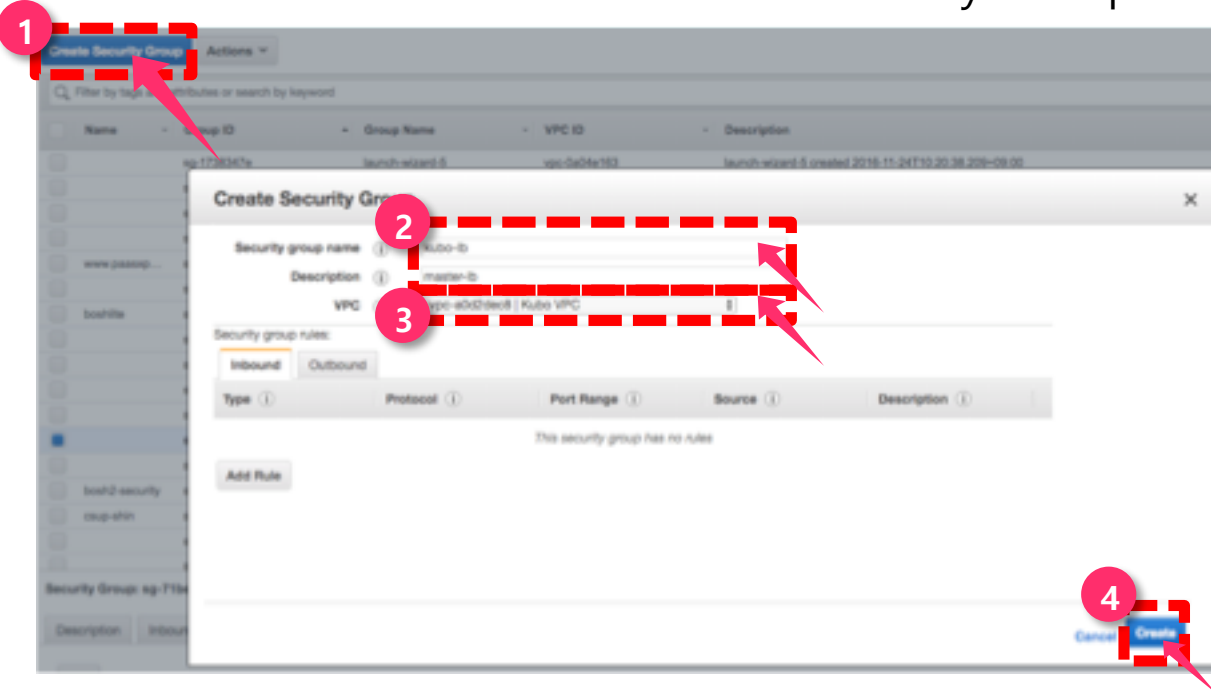
- Make sure to have Same Inbound and Outbound Values.

1. AWS Infrastructure Setting (7/12)

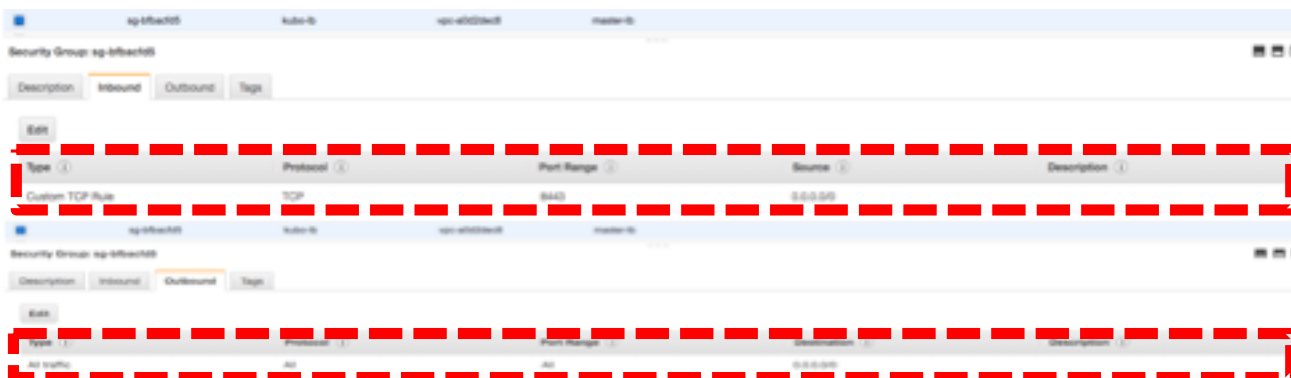
- Create Security Groups (2/2)

❖ Move to Security Group TAB.

* AWS Services → EC2 → Security Groups



- This Security Group is Used only for Kubernetes Master Load Balancer.



- Make sure to have Same Inbound and Outbound Values.

1. AWS Infrastructure Setting (8/12)

- Create Load Balancer (1/3)

❖ Create Classic Load Balancer.

* AWS Services → EC2 → Load Balancers → Classic Load Balancer

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings 4. Configure Health Checks 5. Add EC2 Instances 6. Add Tags 7. Review

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through creating a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. We've configured your load balancer with a standard web server on port 80.

Create Load Balancer Name: (Required) | VPC: (Required) | [View VPC](#)

Create an internal load balancer: ☐ (Optional)

Listener Configuration

Provide advanced VPC configuration: ☐ (Optional)

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
TCP	8443	TCP	80

[Add](#)

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC: (Required) | [View VPC](#)

Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

Available subnets	Availability Zone	Subnet ID	Subnet CIDR	Name
<input type="radio"/>	us-east-1a	subnet-01234567	10.0.0.0/24	Public-Private-subnet

Selected subnets	Availability Zone	Subnet ID	Subnet CIDR	Name
<input checked="" type="radio"/>	us-east-1a	subnet-01234567	10.0.0.0/24	Public-Private-subnet

[Cancel](#) [Next](#) [Assign Security Groups](#)

• Name you LB and Select your VPC where LB want's to create.

• Make sure to listen TCP 8443 Port

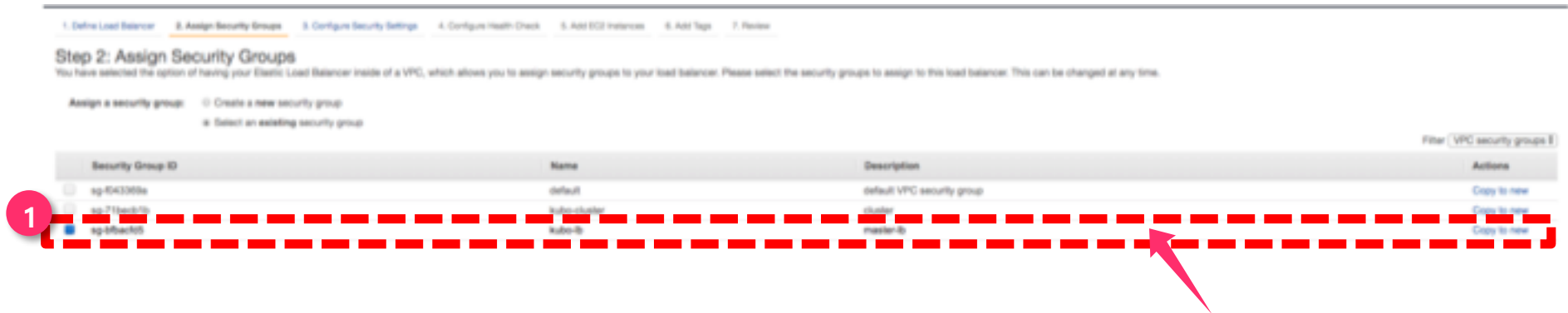
• Make Sure to select public subnet

1. AWS Infrastructure Setting (9/12)

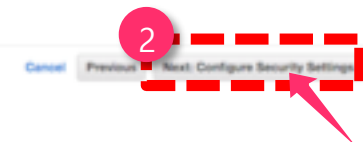
- Create Load Balancer (2/3)

❖ Create Classic Load Balancer.

* AWS Services → EC2 → Load Balancers → Classic Load Balancer



- Make sure to select **kubernetes** security group for security reasons



1. AWS Infrastructure Setting (10/12)

- Create Load Balancer (3/3)

❖ Create Classic Load Balancer.

* AWS Services → EC2 → Load Balancers → Classic Load Balancer

1

Step 7: Review
Please review the load balancer details before continuing.

- Define Load Balancer
Load Balancer name: master-01
Scheme: Internet-facing
Port Configuration: 80 (TCP) forwarding to 80 (TCP)
[Edit load balancer definition](#)
- Configure Health Check
Ping Target: TCP:80
Timeout: 5 seconds
Interval: 30 seconds
Unhealthy threshold: 2
Healthy threshold: 10
[Edit health check](#)
- Add EC2 Instances
Cross-Zone Load Balancing: Enabled
Connection Draining: Enabled, 300 seconds
Instances:
[Edit instances](#)
- VPC Information
VPC: vpc-af01d0e8 (Kube VPC)
Subnets: subnet-c0bb0ea0 (Kube Public subnet)
[Edit subnets](#)
- Security groups
Security groups: sg-0f9a7d01
[Edit security groups](#)

- Make sure to check all the above properties match your requirements.

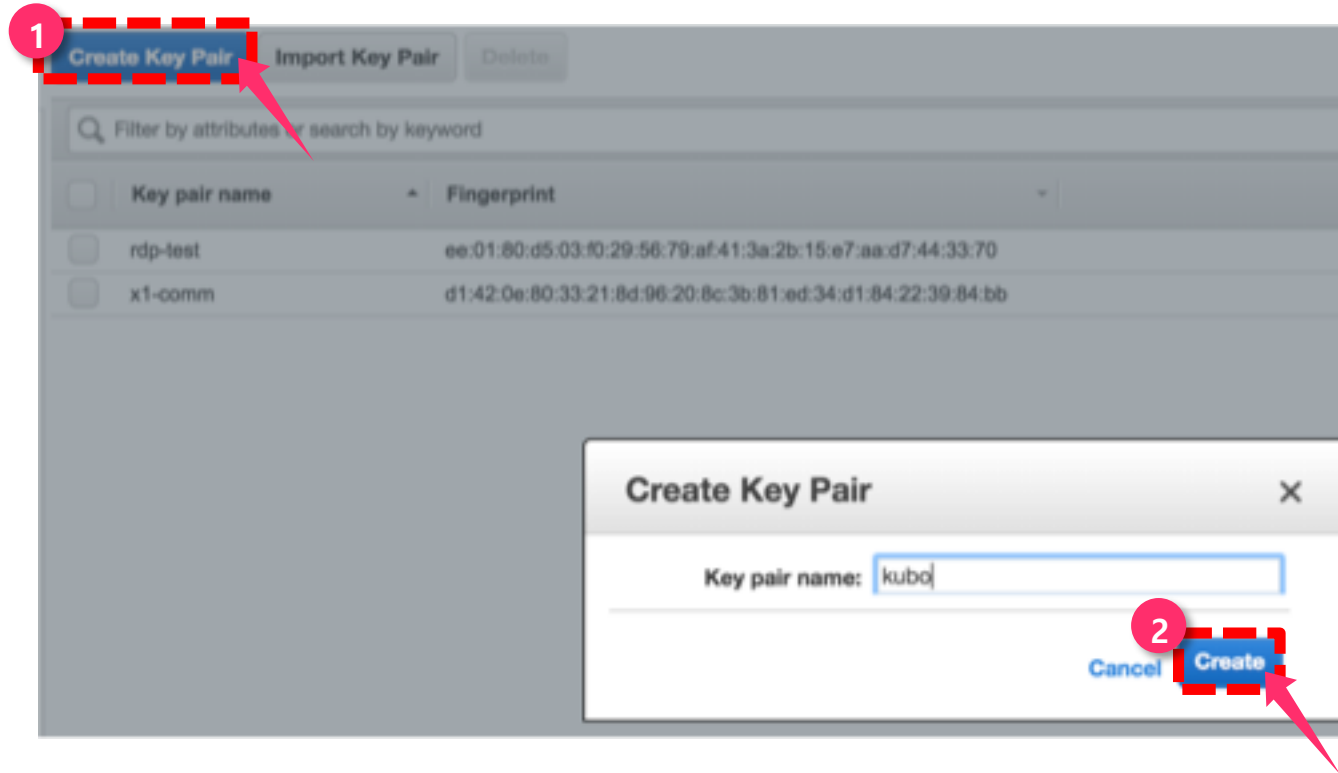
2

1. AWS Infrastructure Setting (11/12)

- Create Keypairs

❖ Create Classic Load Balancer.

* AWS Services → EC2 → Key Pairs → Classic Load Balancer



- Make sure to save key pair in safe place for future proceedings.

1. AWS Infrastructure Setting (12/12)

- Create Instance

❖ Create Classic Load Balancer.

* AWS Services → EC2 → Instances

Requirements	Description
OS	Select Ubuntu 14.04 LTS
Instance Size	Select t2.micro
Network	Select your VPC (Kubo VPC)
Subnet	Select Your Public Subnet (Kubo Public subnet)
Auto-assign Public IP	Manually Enable
Storage	Minimum 40 GB
Security Group	Select kubo-cluster Security Group
Key	Select Key Pair you created before

2. Deploying Bosh Director on AWS (1/8)

- Installing Pre-Requirements (1/3)

Note: Perform below steps in your AWS Inception.

❖ ssh into inception

```
$ ssh ubuntu@your-aws-inception-public-domain
```

❖ Install bosh-cli-2.0

```
$ wget https://s3.amazonaws.com/bosh-cli-artifacts/bosh-cli-3.0.1-linux-amd64
$ chmod +x bosh-cli-3.0.1-linux-amd64
$ sudo mv bosh-cli-3.0.1-linux-amd64 /usr/local/bin/bosh
```

❖ Check the bosh version

```
$ bosh -v
version 3.0.1-712bfd7-2018-03-13T23:26:42Z
```

❖ Check the bosh version

```
$ sudo apt-get install -y build-essential zlibc zlib1g-dev ruby ruby-dev openssl libxslt-dev libxml2-dev libssl-dev libreadline6 libreadline6-dev libyaml-dev libsqlite3-dev sqlite3
```

2. Deploying Bosh Director on AWS (2/8)

- Installing Pre-Requirements (2/3)

❖ Install Git

```
$ sudo apt-get install git  
$ git --version  
git version 2.14.3
```

❖ Install Credhub-Cli

```
$ wget https://github.com/cloudfoundry-incubator/credhub-cli/releases/download/1.7.6/credhub-linux-1.7.6.tgz  
$ tar -xvf credhub-linux-1.7.6.tgz  
$ chmod +x credhub  
$ sudo mv credhub /usr/local/bin/credhub
```

❖ Check the credhub version

```
$ credhub --version  
CLI Version: 1.6.0
```


2. Deploying Bosh Director on AWS (3/8)

- Installing Pre-Requirements (3/3)

❖ Install Kubectl-Cli

```
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.10.3/bin/linux/amd64/kubectl  
$ chmod +x kubectl  
$ sudo mv kubectl /usr/local/bin/kubectl  
$ kubectl version
```

2. Deploying Bosh Director on AWS (4/8)

- Deploying Bosh-Director (1/5)

- ❖ In your Inception

```
$ mkdir ~/workspace && cd ~/workspace
```

- ❖ Next clone the bosh-deployment and kubo-deployment repositories

```
$ git clone https://github.com/cloudfoundry/bosh-deployment.git  
$ git clone https://github.com/cloudfoundry-incubator/kubo-deployment.git -b v0.17.0
```

- ❖ Edit AWS cpi.yml file to configure Nat VM.

```
$ vi ~/workspace/bosh-deployment/aws/cpi.yml
```

Before Editing

```
# Configure AWS sizes  
- type: replace  
  path: /resource_pools/name=vms/cloud_properties?  
  value:  
    instance_type: m4.xlarge  
    ephemeral_disk:  
      type: gp2  
      size: 25_000  
    availability_zone: ((az))
```

After Editing

```
# Configure AWS sizes  
- type: replace  
  path: /resource_pools/name=vms/cloud_properties?  
  value:  
    instance_type: t2.medium  
    ephemeral_disk:  
      type: gp2  
      size: 25_000  
    availability_zone: ((az))
```

Change your instance_type according to your usage

2. Deploying Bosh Director on AWS (5/8)

- Deploying Bosh-Director (2/5)

- ❖ Edit Bosh bosh.yml file to configure Nat VM.

```
$ vi ~/workspace/bosh-deployment/bosh.yml
```

Before Editing

```
disk_pools:
- name: disks
  disk_size: 65_536

networks:
- name: default
  type: manual
  subnets:
  - range: ((internal_cidr))
    gateway: ((internal_gw))
    static: (((internal_ip)))
    dns: [8.8.8.8]
```

After Editing

```
disk_pools:
- name: disks
  disk_size: 40_960

networks:
- name: default
  type: manual
  subnets:
  - range: ((internal_cidr))
    gateway: ((internal_gw))
    static: [((internal_ip))]
    dns: [10.0.1.1]
```

Your Private Subnet CIDR & Nat Gateway

- ❖ Create directory to store director deployment state and credentials.

```
$ cd ~/workspace/bosh-deployment/ && mkdir kubo && cd kubo
```

2. Deploying Bosh Director on AWS (6/8)

- Deploying Bosh-Director (3/5)

- ❖ Below command create bosh-init vm on virtualbox by means of bosh.yml as base manifest.
- ❖ Yaml files with **-o command** set variables **director_name**, **internal_ip**, **internal_gw**, **internal_cidr**, **outbound_network_name** using **-v command**.
- ❖ It also create **state.json** to record running state and **creds.yml** (**for certs & credentails**) in your ~/deployments/vbox directory
- ❖ Following Command Creates Director's vm and install it's jobs

```
$ bosh create-env ~/workspace/bosh-deployment/bosh.yml
--state=~/workspace/bosh-deployment/kubo/state.json
-o ~/workspace/bosh-deployment/aws/cpi.yml
-o ~/workspace/bosh-deployment/jumpbox-user.yml
-o ~/workspace/bosh-deployment/uaa.yml
-o ~/workspace/bosh-deployment/credhub.yml
--vars-store=~/workspace/bosh-deployment/kubo/creds.yml
-v director_name="kubo"
-v internal_ip=10.0.1.6
-v internal_gw=10.0.1.1
-v internal_cidr=10.0.1.0/24
-v access_key_id=Your IAM User Access Key
-v secret_access_key=Your IAM User Secret Access Key
-v region=ap-northeast-2
-v az= ap-northeast-2a
-v default_key_name=Your Key Name Created Before (kubo)
-v default_security_groups= Your Secirity Group Name Created Before (kubo-cluster)
--var-file private_key=~ /kubo.pem (Copy your key to your inception home director)
-v subnet_id=subnet-6ebabf06 (Your Private Subnet in Your VPC)
```

2. Deploying Bosh Director on AWS (7/8)

- Deploying Bosh-Director (4/5)

- ❖ Log into director with your alias name.

```
$ bosh alias-env kubo -e 10.0.1.6 --ca-cert <(bosh int ~/workspace/bosh-deployment/kubo/creds.yml --path /director_ssl/ca)

$ export BOSH_CLIENT=admin

$ export BOSH_CLIENT_SECRET=`bosh int ~/workspace/bosh-deployment/kubo/creds.yml --path /admin_password`
```

- ❖ Create directory to download releases and stemcells

```
$ mkdir ~/workspace/releases && cd ~/workspace/releases
```

- ❖ Download Upload AWS Stemcell

```
$ wget https://s3.amazonaws.com/bosh-core-stemcells/aws/bosh-stemcell-3586.16-aws-xen-hvm-ubuntu-trusty-go_agent.tgz
$ bosh -e kubo upload-stemcell bosh-stemcell-3586.16-aws-xen-hvm-ubuntu-trusty-go_agent.tgz
```

2. Deploying Bosh Director on AWS (8/8)

- Deploying Bosh-Director (5/5)

❖ Target & Log into director credhub

```
$ export CREDHUB_CLIENT=credhub-admin  
$ export CREDHUB_SECRET=$(bosh int --path /credhub_admin_client_secret ~/workspace/bosh-deployment/kubo/creds.yml)  
$ export CREDHUB_CA_CERT=$(bosh int --path /credhub_tls/ca ~/workspace/bosh-deployment/kubo/creds.yml)  
$ credhub login -s https://10.0.1.252:8844 --skip-tls-validation
```

❖ List out Certificates and Passwords in credhub

```
$ credhub find
```

❖ To Delete Certificates and Passwords in credhub

```
$ credhub delete -n /director_name/deployment-name/certificate-name
```

3. Deploying KUBO on AWS (1/8)

- Deploying Kubo (1/5)

- ❖ Download kubo-release.

```
$ cd ~/workspace && mkdir releases && cd ~/workspace/releases  
$ wget https://github.com/cloudfoundry-incubator/kubo-release/releases/download/v0.17.0/kubo-release-0.17.0.tgz
```

- ❖ Upload kubo-release

```
$ bosh -e kubo upload-release ~/workspace/releases/kubo-release-0.17.0.tgz
```

- ❖ Edit kubo-deployment cfc.yml file for deploying kubernetes master bosh-lite

```
$ cd ~/workspace/kubo-deployment  
$ vi ~/workspace/kubo-deployment/manifests/cfc.yml
```

Before Editing

```
77 - name: master  
78   instances: 3  
79   networks:  
80     - name: default  
81   azs: [z1,z2,z3]
```

After Editing

```
69 - name: master  
70   instances: 1  
71   networks:  
72     - name: default  
73   azs: [z1]
```

3. Deploying KUBO on AWS (2/8)

- Deploying Kubo (2/5)

- ❖ Edit kubo-deployment cfcf.yml file for deploying Kubernetes worker-nodes AWS

```
$ vi ~/workspace/kubo-deployment/manifests/cfcf.yml
```

Before Editing

```
155 - name: worker
156   instances: 3
157   networks:
158     - name: default
159     azs: [z1,z2,z3]
```

After Editing

```
- name: worker
  instances: 2
  networks:
    - name: default
    azs: [z1]
```

- ❖ Edit kubo-deployment cfcf.yml file for deploying kubernetes master certificates

```
$ vi ~/workspace/kubo-deployment/manifests/cfcf.yml
```

Before Editing

```
237 - name: tls-kubernetes
238   type: certificate
239   options:
240     ca: kubo_ca
241     organization: "system:masters"
242     common_name: master.cfcr.internal
243     alternative_names:
244       - 10.100.200.1
245       - kubernetes
246       - kubernetes.default
247       - kubernetes.default.svc
248       - kubernetes.default.svc.cluster.local
249       - master.cfcr.internal
```

After Editing

```
- name: tls-kubernetes
  type: certificate
  options:
    ca: kubo_ca
    organization: "system:masters"
    common_name: cfcr-cfcr-api-980222485.ap-northeast-2.elb.amazonaws.com
    alternative_names:
      - 10.100.200.1
      - kubernetes
      - kubernetes.default
      - kubernetes.default.svc
      - kubernetes.default.svc.cluster.local
      - master.cfcr.internal
      - cfcr-cfcr-api-980222485.ap-northeast-2.elb.amazonaws.com
```

Note: Change cfcr-cfcr-api-980222485.ap-northeast-2.elb.amazonaws.com according to your LB DNS Name Created in Previous Steps.

3. Deploying KUBO on AWS (3/8)

- Deploying Kubo (3/5)

- ❖ Create and Save bosh director cloud-config

```
$ vi ~/workspace/kubo-deployment/manifests/cloud-config.yml
```

```
ops:
- cloud_properties:
  availability_zone: ap-northeast-2a
  name: z1
- cloud_properties:
  availability_zone: ap-northeast-2a
  name: z2
- cloud_properties:
  availability_zone: ap-northeast-2a
  name: z3
compilation:
  ac: z1
  network: default
  reuse_compilation_vms: true
  vm_type: worker
  workers: 3
disk_types:
- cloud_properties:
  encrypted: true
  type: gp2
  disk_size: 5120
  name: 5120
- cloud_properties:
  encrypted: true
  type: gp2
  disk_size: 10240
  name: 10240
networks:
- name: default
  subnets:
  - ops:
    - z1
    - z2
    - z3
    cloud_properties:
      subnet: subnet-c45451ac
    dns:
      - 10.0.1.1
    gateway: 10.0.1.1
    range: 10.0.1.0/24
    reserved:
      - 10.0.1.1/30
    type: manual
  vm_types:
  - cloud_properties:
    ephemeral_disk:
      size: 25000
    instance_type: t2.small
    name: minimal
  - cloud_properties:
    elbs:
      - cfc-cfcr-api
    ephemeral_disk:
      size: 25000
    instance_type: t2.small
    name: small
  - cloud_properties:
    ephemeral_disk:
      size: 60000
    instance_type: t2.medium
    name: small-highmem
```

3. Deploying KUBO on AWS (4/8)

- Deploying Kubo (4/5)

- ❖ Update bosh director cloud-config

```
$ bosh -e kubo update-cloud-config ~/workspace/kubo-deployment/manifests/cloud-config.yml
```

- ❖ Check the desired releases and stemcell to deploy kubo on AWS bosh

```
$ bosh -e kubo releases
```

Output

```
Using environment '10.0.1.252' as client 'admin'
```

Name	Version	Commit Hash
bosh-dns	1.5.0*	f5a8d25
bpm	0.6.0*	b6f4675
cfc-etc	1.3*	6a62d8f
docker	32.0.0*	542c382
kubo	0.17.0*	ad9ef809

```
(*) Currently deployed  
(+) Uncommitted changes
```

```
5 releases
```

```
Succeeded
```

```
$ bosh -e kubo stemcells
```

3. Deploying KUBO on AWS (5/8)

- Deploying Kubo (5/5)

❖ Deploy Kubo on AWS

```
$ bosh -e kubo -d cfcr deploy ~/workspace/kubo-deployment/manifests/cfcr.yml -o ~/workspace/kubo-deployment/manifests/ops-files/iaas/aws/cloud-provider.yml -o ~/workspace/kubo-deployment/manifests/ops-files/iaas/aws/add-master-credentials.yml -o ~/workspace/kubo-deployment/manifests/ops-files/iaas/aws/add-worker-credentials.yml -v aws_access_key_id_master=XXXX -v aws_secret_access_key_master=XXXX -v aws_access_key_id_worker=XXXX -v aws_secret_access_key_worker=XXXX -o ~/workspace/kubo-deployment/manifests/ops-files/iaas/aws/lb.yml -v kubernetes_cluster_tag=kubocrxpert
```

❖ Deploy Kubernetes add-ons

```
$ bosh -e kubo -d cfcr run-errand apply-specs
```

❖ Check Kubernetes Deployment

```
$ bosh -e kubo -d cfcr vms
```

Output

```
Using environment '10.0.1.252' as client 'admin'
```

```
Task 141. Done
```

```
Deployment 'cfcr'
```

Instance	Process State	AZ	IPs	VM CID	VM Type	Active
master/528ffe75-5391-4313-8ec6-6c1941a17685	running	z1	10.0.1.101	vm-befdcalf-ebdd-4018-44d8-feb6bc8ab462	small	true
worker/61b51092-6a00-4c74-8c10-e3e6b479487e	running	z1	10.0.1.102	vm-e48278ef-3225-4860-4bfa-efb283438085	small-highmem	true
worker/bd7caa22-39c1-40f9-bc8a-efe3471220ca	running	z1	10.0.1.103	vm-c8582161-1077-41f4-7050-22febe004840	small-highmem	true

```
3 vms
```

```
Succeeded
```

❖ ssh into master and work vms

```
$ bosh -e kubo -d cfcr ssh master
```

```
$ bosh -e kubo -d cfcr ssh worker/61b51092-6a00-4c74-8c10-e3e6b479487e
```

3. Deploying KUBO on AWS (6/8)

- Accessing Kubernetes (1/3)

- ❖ Download tls-kubernetes certificate from credhub

```
$ bosh -e kubo -d cfcf int <(credhub get -n "/kubo/cfcf/tls-kubernetes" --output-json) --path=/value/ca > ~/workspace/kubo-deployment/kubo/kubernetes.crt
```

- ❖ Download kubo-admin-password password from credhub

```
$ bosh -e kubo -d cfcf int <(credhub get -n "/kubo/cfcf/kubo-admin-password" --output-json) --path=/value > ~/workspace/kubo-deployment/kubo/kubernetes_pwd.crt
```

- ❖ Configure Kubernetes Cluster by using above Certificate (kubernetes.crt)

```
$ kubectl config set-cluster "dev" --server https://you-lb-dns:8443 --embed-certs=true --certificate-authority=~/workspace/kubo-deployment/kubo/kubernetes.crt
```

- ❖ Configure Kubernetes User and Context (Note: Use Kubernetes User Password from above kubernetes_pwd.crt)

```
$ kubectl config set-credentials "dev" --token=your-kubernetes_pwd.crt-password
```

```
$ kubectl config set-context "dev" --cluster="dev" --user="dev"
```

```
$ kubectl config use-context "dev"
```

3. Deploying KUBO on AWS (7/8)

- Accessing Kubernetes (2/3)

❖ Obtain kubernetes node Information

```
$ kubectl get node -o wide
```

Output

NAME	STATUS	ROLES	AGE	VERSION	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
vm-c8582161-1877-41f4-7858-22feb8084848	Ready	<none>	21h	v1.18.3		Ubuntu 14.04.5 LTS	4.4.0-127-generic	docker://17.12.1-ce
vm-e48278ef-3225-4868-4bfa-efb283438885	Ready	<none>	21h	v1.18.3		Ubuntu 14.04.5 LTS	4.4.0-127-generic	docker://17.12.1-ce

❖ Obtain kubernetes Cluster Information

```
$ kubectl cluster-info
```

❖ Get all information regarding your Namespace

```
$ kubectl get pods --namespace=kube-system  
$ kubectl get all -n kube-system
```

❖ Check 8001 port is active on your notebook, if active kill the process

```
$ sudo lsof -PiTCP -sTCP:LISTEN  
$ sudo kill -9 your-pid-number
```

❖ Accessing to Kubernetes Dashboard

```
$ kubectl proxy  
Starting to serve on 127.0.0.1:8001  
$ http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login
```

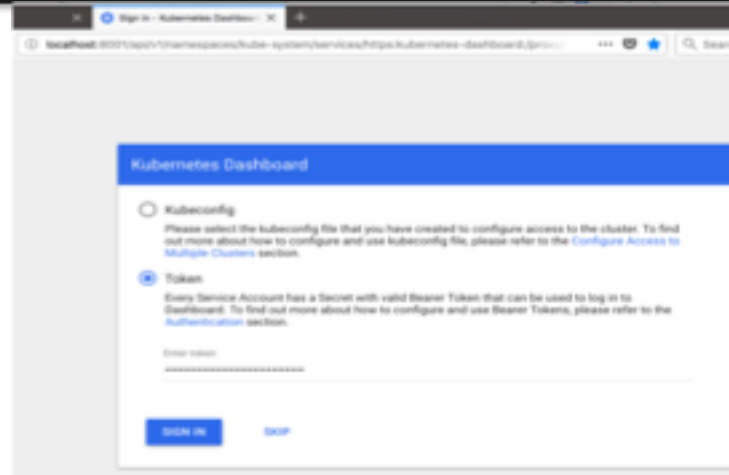
3. Deploying KUBO on AWS (8/8)

- Accessing Kubernetes (3/3)

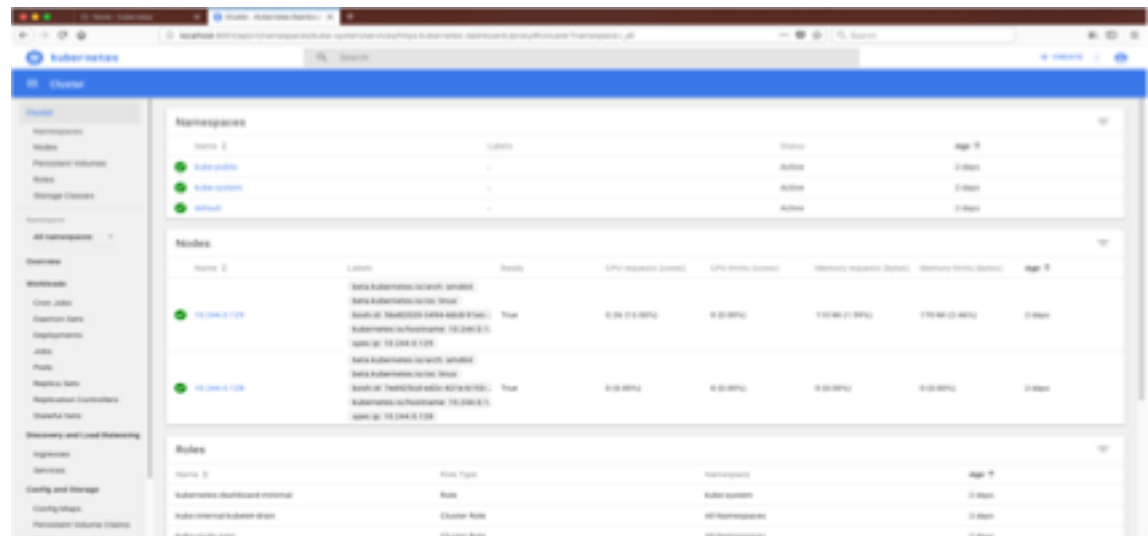
❖ Accessing Kubernetes Dashboard in your Browser

Select token option and paste above kubernetes_pwd.crt password to login into your kubernetes cluster

Output

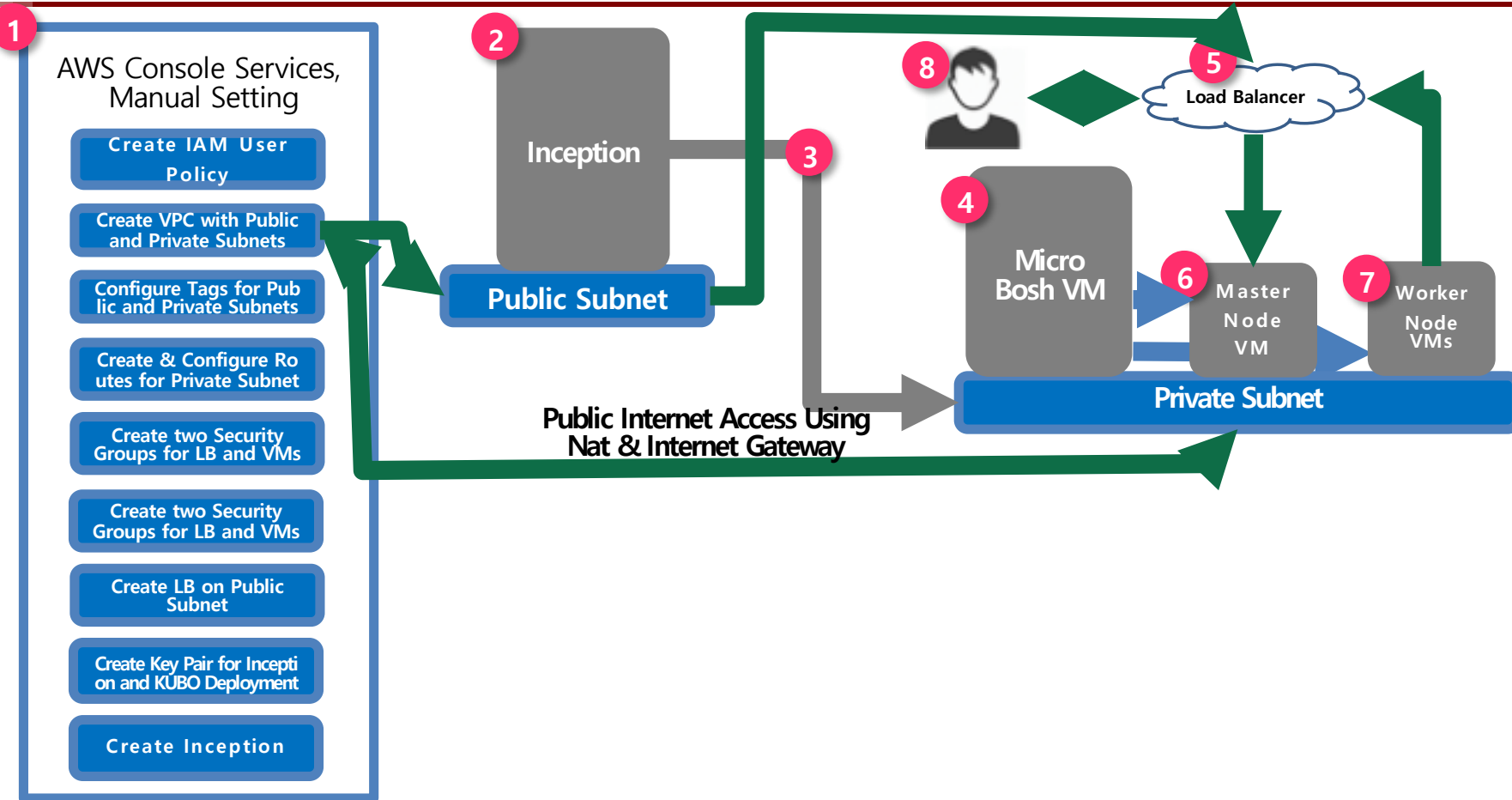


❖ Kubernetes Home Dashboard Output



4. KUBO Deployment Architecture on AWS

- Deployment Architecture



THANK YOU