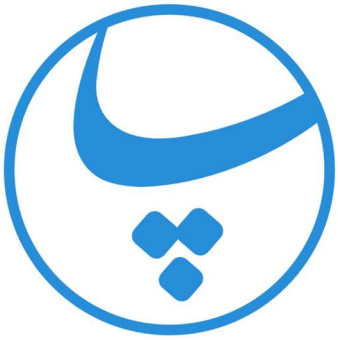

# PARSPACK.COM

## آموزش نصب کلاسترینگ کوبرنیتیس Kubernetes 1.10 با Kubeadm در لینوکس Ubuntu 16.04

## How To Create a Kubernetes 1.10 Cluster Using Kubeadm on Ubuntu 16.04

### فهرست

## معرفی

کوبرنیتیس Kubernetes یک سیستم مدیریت مخازن است که به صورت متن باز و رایگان می باشد. این پروژه توسط گوگل و بر اساس تجربیات اجرای مخازن در محصولات ایجاد شده است. در این آموزش به نحوه نصب ، راه اندازی و تنظیمات کلاسترینگ Kubernetes بر روی لینوکس Ubuntu می پردازیم. این آموزش یک آموزش حرفه ای است و برای نصب Kubernetes از Kubeadm و برای تنظیمات سرور های موجود در کلاستر از Ansible استفاده می کنیم.

**مشاهده آنلاین این آموزش در سایت پارس پک**

**در این آموزش با توضیحات و جزئیات کامل و درک همه مفاهیم به نصب Kubernetes در لینوکس Ubuntu می پردازیم ، اگر مایل هستید که فقط دستورات و آموزش نصب را به صورت خلاصه برای نصب سریع داشته باشید به لینک زیر مراجعه کنید.**

**مشاهده خلاصه آموزش و دستورات این آموزش**

از Kubeadm برای نصب ، پیکربندی و خودکارسازی اجزای Kubernetes مانند API های سرور ، Controller Manager و Kube DNS استفاده می شود. این برنامه کارهایی مثل ایجاد کاربر یا مدیریت مراحل نصب سیستم عامل و تنظیمات آن ها را انجام نمی دهد و برای این کار ها ممکن از ابزارهای مدیریت پیکربندی مانند Ansible یا SaltStack استفاده شود. از ابزار Kubeadm ایجاد کلاستر ها یا بازسازی آنها به سادگی و با کمترین خطا استفاده می شود.

## اهداف

در این آموزش برای کلاسترینگ از سرور های زیر استفاده می کنیم :

یک سرور برای Master Node

این سرور مسئولیت مدیریت کلاستر را دارد. در این سرور Etcd اجرا می شود که اطلاعات کلاستر مربوط به تقسیم بار کاری بین سرور ها را ذخیره می کند.

دو سرور برای Worker Node ها

سرور هایی هستند که کلاستر و تقسیم بار بر روی آنها انجام می شود( مانند برنامه ها و سرویس های مخازن ها ). وظایف Worker ها توسط سرور Master تقسیم بندی و زمان بندی شده و با آن ها داده می شود ، و یک سرور worker ، وظیفه ای که به آن داده شده است را تا زمانی که Master زمان بندی کرده است انجام می دهد. با اضافه کردن سرور های worker می توان ظرفیت کلاستر را افزایش داد.

بعد از اتمام این آموزش یک کلاستر برای اجرای برنامه های مخازن داریم که منابع RAM و CPU مورد نیاز برای اجرای برنامه ها را با سرور های متعدد تامین می کند. تقریبا تمامی برنامه ها و سرویس های مبتنی بر لینوکس ، مانند برنامه های تحت وب ، پایگاه داده ، سرویس ها و ابزارهای دستورات

خط فرمان ، در کلاسترینگ Kubernetes قابل اجرا هستند. خود کلاستر نیز حدود ۵۰۰-۳۰۰ MB از حافظه RAM و ۱۰ درصد از CPU را مصرف می کند.

در این آموزش وقتی کلاستر راه اندازی راه اندازی کردیم ، یک وب سرور Nginx را نصب می کنیم تا مطمئن شویم که همه چیز درست است.

## پیش نیاز ها

کلید SSH که بر روی هر دو سرور محلی تنظیم شده است. این کلید در تنظیمات سرور ها برای برقرار ارتباط بین سرور ها قرار می گیرد و اگر تا الان با کلید های SSH کار نکرده اید از این آموزش می توانید استفاده کنید.

سه سرور Ubuntu16.04 با حداقل یک گیگ رم که تنظیمات SSH با کلید روی آن انجام شده باشد.

برنامه Ansible بر روی سرورهای مجازی محلی نصب شده باشد. برای آشنایی با نصب Ansible صفحه آموزش نصب و پیکربندی Ansible در Ubuntu را مطالعه کنید.

آشنایی با Ansible Playbooks ها . برای آشنایی با Playbook های Ansible صفحه آموزش مدیریت پیکربندی و نوشتن Playbook های Ansible را مطالعه کنید.

دانش راه اندازی یک مخزن با داکر Docker. مرحله پنجم از آموزش نصب داکر در اوبونتو را مرور کنید.

## مرحله اول – تنظیم دایرکتوری فضای کار و فایل فهرست Ansible

در این مرحله یک دایرکتوری برای سرویس دهی فضای کاری یا همان Workspace ایجاد می کنیم و تنظیمات Ansible را به صورت محلی یا Local انجام می دهیم تا بتوانیم با سرور ها به صورت ارتباط راه دور یا Remote ارتباط برقرار کنیم و دستورات خط فرمان را اجرا کنیم. برای انجام این کار یک فایل با نام hosts که شامل اطلاعات سرورها از قبیل آدرس IP ها و گروه های سرور های هستند را ایجاد می کنیم.

در این آموزش سه سرور داریم که یکی از این سه سرور ، سرور master مان است که IP آن را با master_ip نشان می دهیم و آدرس IP دو سرور دیگر که به عنوان worker هستند را با نام های worker_1_ip و worker_2_ip نمایش می دهیم.

یک دایرکتوری با نام kube-cluster/~ در مسیر home ایجاد کنید و وارد آن شوید :

```
mkdir ~/kube-cluster
cd ~/kube-cluster
```

این دایرکتوری ، فضای کاری ما یا همان Workspace در این آموزش خواهد بود و شامل playbook های Ansible است و تمام دستورات خط فرمان مربوط به این سرور را در این دایرکتوری وارد می کنیم.

ایجاد یک فایل با نام kube-cluster/hosts/~ با ابزار nano یا ویرایشگر متن مورد نظرتان :

```
nano ~/kube-cluster/hosts
```

خطوط زیر را در این فایل وارد کنید و مقادیر گفته شده را با مقادیر مناسب جایگزین کنید. منظور مقادیر master_ip و worker_1_ip و worker_2_ip می باشد :

```
[asters]
master ansible_host=master_ip ansible_user=root
[workers]
worker1 ansible_host=worker_1_ip ansible_user=root
worker2 ansible_host=worker_2_ip ansible_user=root
[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

اطلاعات این فایل ، دارایی ها یا همان اطلاعات سرور ها خواهند بود که شامل اطلاعاتی از قبیل آدرس IP ، نام کاربری کاربرانی که از راه دور به صورت Remote متصل می شوند و نام گروه های سرور ها می باشد ، که در اینجا شامل گروه سرور های masters و workers می باشد که از این فایل و اطلاعات در قسمت های دیگر استفاده می کنیم.

در گروه masters ، یک سرور وجود دارد که با نام master مشخص شده با آدرس IP و نام کاربر راه دور که Ansible باید برای اجرای دستورات خود را از آن استفاده کند که در اینجا کاربر root مشخص شده است.

به همین ترتیب ، در گروه workers ، دو سرور worker با آدرس های worker_1_ip و worker_2_ip و کاربر ansible_user مشخص شده اند.

خط آخر نیز برای مدیریت عملیات ها از راه دور است ، برنامه Ansible مشخص شده است که از مترجم Python 3 در سرور ها استفاده کند.

در انتها نیز بعد از ویرایش فایل و اطمینان از صحت اطلاعات آن ، فایل را ذخیره کرده و خارج شوید.

تا اینجا لیست مشخصات سرور ها را تکمیل کردیم ، در ادامه مراحل نصب به موارد مورد نیاز سیستم عامل و پیکربندی تنظیمات می پردازیم.

## مرحله دوم – ایجاد کاربر در سرور های متصل

در این قسمت بر روی سرور ها یک کاربر عادی بدون دسترسی ریشه non-root user ایجاد می کنیم ولی این کاربر ها باید عضو گروه sudo باشند تا بتوان بوسیله این کاربر ها از طریق اتصال SSH با سرور ارتباط برقرار کرد. این کار برای زمان هایی که می خواهید اطلاعات سیستم را با دستوراتی مانند top/htop مشاهده کنید و لیستی از منابع در حال اجرا را مشاهده کنید یا تغییراتی را در تنظیمات فایل هایی که مالک آن root است ، تغییر دهید بسیار مفید است.

هر چند که این کارها را به صورت مرتب در طول نگهداری کلاستر انجام خواهیم داد ، و استفاده از یک کاربر عادی non-root ریسک عملیات هایی مانند ویرایش یا حذف فایل ها و عملیات هایی که ناخواسته انجام می شود را کاهش می دهد.

یک فایل با نام kube-cluster/initial.yml~/ در فضای کاری Workspace ایجاد کنید :

```
nano ~/kube-cluster/initial.yml
```

در ادامه متن زیر را که play نامیده می شود، در فایل قرار دهید تا کاربر مورد نظر در تمام سرور ها ایجاد شود. یک play در Ansible مجموعه ای از مراحلی است که مشخصات سرور ها و گروه های مشخصی را تهیه می کند. Play زیر یک کاربر non-root ایجاد می کند :

```
-hosts: all
 become: yes
 tasks:
 - name: create the 'ubuntu' user
   user: name=ubuntu append=yes state=present createhome=yes shell=/bin/bash
 - name: allow 'ubuntu' to have passwordless sudo
   lineinfile:
    dest: /etc/sudoers
    line: 'ubuntu ALL=(ALL) NOPASSWD: ALL'
    validate: 'visudo -cf %s'
 - name: set up authorized keys for the ubuntu user
   authorized_key: user=ubuntu key="{{item}}"
   with_file:
./~ -      ssh/id_rsa.pub
```

توضیحات اینکه این play چکار می کند :

- ایجاد یک کاربر non-root با نام ubuntu
- تنظیم فایل sudoers برای اجازه اجرای دستورات sudo بدون درخواست رمز عبور برای کاربر ubuntu
- اضافه کردن کلید عمومی ماشین محلی به لیست کلید های مجاز کاربر ریموت ubuntu. با این کار اجازه اتصال SSH به هر سروری داده می شود.

این فایل را ذخیره کرده و خارج شوید.

در ادامه این playbook را اجرا می کنیم :

```
ansible-playbook -i hosts ~/kube-cluster/initial.yml
```

این دستور بین دو تا ۵ دقیقه اجرا می شود ، و پس از کامل اجرا شدن دستور خروجی مشابه زیر نمایش داده می شود :

```
Output
PLAY [all]****

TASK [Gathering Facts]****

ok: [master]

ok: [worker1]

ok: [worker2]

TASK [create the 'ubuntu' user]****

changed: [master]

changed: [worker1]

changed: [worker2]

TASK [allow 'ubuntu' user to have passwordless sudo]****

changed: [master]

changed: [worker1]

changed: [worker2]

TASK [set up authorized keys for the ubuntu user]****

changed: [worker1] => (item=ssh-rsa AAAAB3…

changed: [worker2] => (item=ssh-rsa AAAAB3…

changed: [master] => (item=ssh-rsa AAAAB3…

PLAY RECAP****

master                     : ok=5 changed=4 unreachable=0    failed=0

worker1                    : ok=5 changed=4 unreachable=0    failed=0

worker2                    : ok=5 changed=4 unreachable=0    failed=0
```

تا اینجا تنظیمات مقدماتی انجام شده و می توانید به نصب پیش نیاز های کوبرنیتیس Kubernetes بپردازید.

## مرحله سوم – نصب پیش نیاز های kubernetes

در این قسمت باید پکیج های مورد نیاز Kubernetes در سیستم عامل را بوسیله apt که ابزار مدیریت بسته های Ubuntu است ، نصب کنیم. این پکیج ها به صورت زیر هستند :

**Doker** - یک مخزن زمان اجرا می باشد، که شامل اجزای مورد نیاز مخازن شما می باشد. همچنین مخازن دیگر مانند rkt را پشتیبانی می کند.

**kubeadm** – یک ابزار خط فرمان برای نصب و پیکربندی اجزای مختلف کلاستر به صورت استاندارد می باشد.

**kubelet** - یک برنامه سیستمی و یک سرویس است که برای مدیریت گره ها Nodes و عملیات ها می باشد.

**kubectl** - یک ابزار خط فرمان برای ارسال دستورات به کلاستر ها از طریق API Server می باشد.

نصب این بسته ها به صورت دستی و یکی یکی بر روی همه سرورهای کلاستر زمانبر است و به همین دلیل از Ansible برای این کار استفاده می کنیم و برای این کار باید یک Playbook دیگر تنظیم کنیم.

یک فایل با نام ~/kube-cluster/kube-dependencies.yml در مسیر فضای کاری ایجاد کنید :

```
nano ~/kube-cluster/kube-dependencies.yml
```

سپس play زیر را برای نصب پکیج ها در سرور ها وارد کنید :

```
- hosts: all

  become: yes

  tasks:

    - name: install Docker

      apt:

        name: docker.io

        state: present

        update_cache: true

    - name: install APT Transport HTTPS
```

```
  apt:
    name: apt-transport-https
    state: present
- name: add Kubernetes apt-key
  apt_key:
    url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
    state: present
- name: add Kubernetes' APT repository
  apt_repository:
    repo: deb http://apt.kubernetes.io/ kubernetes-xenial main
    state: present
    filename: 'kubernetes'
- name: install kubelet
  apt:
    name: kubelet
    state: present
    update_cache: true
- name: install kubeadm
  apt:
    name: kubeadm
    state: present
-hosts: master
 become: yes
 tasks:
- name: install kubectl
  apt:
    name: kubectl
    state: present
```

این play به صورت زیر کار می کند :

- نصب داکر
- نصب apt-transport-https ، اضافه کردن منابع HTTPS خارجی به لیست منابع APT
- اضافه کردن apt-key برای تایید کلید منابع Kubernetes APT
- نصب kubelet و kubeadm
- دومین play فقط یک وظیفه را انجام می دهد و آن هم نصب kubectl بر روی سرور master می باشد.

بعد از اتمام کار فایل را ذخیره کرده و خارج شوید و Playbook را اجرا کنید :

```
ansible-playbook -i hosts ~/kube-cluster/kube-dependencies.yml
```

بعد از اتمام اجرای این فایل خروجی مشابه زیر را مشاهده می کنید :

```
Output
PLAY [all]****

TASK [Gathering Facts]****

ok: [worker1]

ok: [worker2]

ok: [master]

TASK [install Docker]****

changed: [master]

changed: [worker1]

changed: [worker2]

TASK [install APT Transport HTTPS]*****

ok: [master]

ok: [worker1]

changed: [worker2]

TASK [add Kubernetes apt-key]*****

changed: [master]

changed: [worker1]

changed: [worker2]

TASK [add Kubernetes' APT repository]*****

changed: [master]

changed: [worker1]
```

```
changed: [worker2]
TASK [install kubelet]*****
changed: [master]
changed: [worker1]
changed: [worker2]
TASK [install kubeadm]*****
changed: [master]
changed: [worker1]
changed: [worker2]
PLAY [master]*****
TASK [Gathering Facts]*****
ok: [master]
TASK [install kubectl]******
ok: [master]
PLAY RECAP****
master                     : ok=9 changed=5 unreachable=0    failed=0
worker1                    : ok=7 changed=5 unreachable=0    failed=0
worker2                    : ok=7 changed=5 unreachable=0    failed=0
```

بعد از نصب داکر ، kubeadm و kubelet بر روی سرور ها ، برای اجرای kubectl چیز خاصی مورد نیاز نیست و فقط باید دستورات کلاستر را اجرا کنید. این مسئله خیلی مهم و حساس است ، که kubectl فقط بر روی سرور master نصب شده باشد و دستورات kubectl فقط از سرور master داده شود. به این نکته دقت کنید که kubectl بر روی هر ماشین مجازی که در کلاستر باشد می تواند نصب شود و دستورات کلاستر را اجرا کند.

خوب تا اینجا پیش نیاز ها نصب شده اند و در ادامه به سراغ نصب و تنظیم سرور master و راه اندازی اولیه کلاستر می رویم.

## مرحله چهارم – تنظیم سرور Master

در این مرحله تنظیمات سرور master را انجام می دهیم. قبل از شروع تنظیمات و ایجاد playbook های جدید کمی درباره برخی اصطلاحات مانند Pods و Pods Networking Plugins صحبت می کنیم. برای آشنایی با Pod ها مطلب مفهوم Pod در kubernetes چیست و چطور کار می کند را مطالعه کنید.

هر Pod آدرس IP برای خود دارد ، و هر pod در هر سرور باید بتواند از طریق آدرس IP آن pod در سرور دیگر به آن متصل شود. مخازنی که بر روی یک سرور هستند براحتی بوسیله رابط های محلی ارتباط برقرار می کنند. ارتباط بین pod ها پیچیده است و نیاز به یک شبکه جداگانه دارد که مسیر شفاف از یک pod به یک pod در یک سرور دیگر مشخص شده باشد.

این کار توسط پلاگین شبکه pod انجام می شود. در این آموزش از Flannel استفاده می کنیم که یک گزینه پایدار و مناسب است.

یک Ansible playbook با نام master.yml ایجاد کنید :

```
nano ~/kube-cluster/master.yml
```

متن play زیر را برای نصب Flannel و مقدار دهی اولیه کلاستر وارد کنید :

```
-hosts: master
 become: yes
 tasks:
 - name: initialize the cluster
   shell: kubeadm init --pod-network-cidr=10.244.0.0/16 >> cluster_initialized.txt
   args:
    chdir: $HOME
    creates: cluster_initialized.txt
 - name: create .kube directory
   become: yes
   become_user: ubuntu
   file:
    path: $HOME/.kube
    state: directory
    mode: 0755
 - name: copy admin.conf to user's kube config
   copy:
    src: /etc/kubernetes/admin.conf
    dest: /home/ubuntu/.kube/config
    remote_src: yes
    owner: ubuntu
 - name: install Pod network
```

```
  become: yes

  become_user: ubuntu

  shell: kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml >> pod_network_setup.txt

  args:

   chdir: $HOME

   creates: pod_network_setup.txt
```

توضیحاتی درباره این play :

- اولین کاری که انجام می شود ، مقدار دهی اولیه کلاستر بوسیله kubeadm init است. تنظیم پارامتر --pod-network-cidr=10.244.0.0/16 ، که شبکه اختصاصی را برای pod مشخص می کند.
- دومین کاری که انجام می شود، ایجاد دایرکتوری .kube در home/ubuntu/ است. در این دایرکتوری اطلاعات پیکربندی از قبیل فایل های کلید ادمین که برای اتصال کلاستر ها هستند و آدرس IP کلاستر ها نگهداری می شود.
- سومین کاری که انجام می شود، کپی فایل /etc/kubernetes/admin.conf که ubeadm init این فایل را با کاربر non-root در home ایجاد کرده است. این کار برای اجازه دسترسی kubectl به کلاستر های جدید ایجاد شده می باشد.
- آخرین کار نیز اجرای kubectl apply برای نصب Flannel می باشد. مقدار kubectl apply -f descriptor.[yml|json] به kubectl اعلام می کند که یک شئ جدید که در فایل descriptor.[yml|json] توضیح داده شده است را ایجاد کند. فایل kube-flannel.yml شامل جزئیات شئ مورد نیاز برای تنظیم Flannel در کلاستر می باشد.

در انتها فایل را ذخیره کرده و خارج شوید.

فایل playbook را به صورت محلی Local اجرا کنید :

```
ansible-playbook -i hosts ~/kube-cluster/master.yml
```

خروجی این دستور مشابه زیر است :

```
Output
PLAY [master]****

TASK [Gathering Facts]****

ok: [master]

TASK [initialize the cluster]****

changed: [master]

TASK [create .kube directory]****

changed: [master]
```

```
TASK [copy admin.conf to user's kube config]*****
changed: [master]
TASK [install Pod network]*****
changed: [master]
PLAY RECAP****
master                     : ok=5 changed=4 unreachable=0    failed=0
```

برای بررسی وضعیت سرور master از طریق ssh با دستور زیر به سرور متصل شوید :

```
ssh ubuntu@master_ip
```

پس از وارد شدن به سرور دستور زیر را اجرا کنید :

```
kubectl get nodes
```

باید خروجی مشابه زیر را مشاهده کنید :

```
Output
NAME      STATUS ROLES     AGE VERSION
master    Ready master    1d v1.10.1
```

در سرور master تمامی تنظیمات و مقداردهی های اولیه انجام شده و وضعیت سرور در حالت ready و آماده برای شروع اتصال به سرور های worker و ارسال وظایف به آنها از طریق API Server می باشد. اکنون می توانیم سرور های worker را اضافه کنیم.

## مرحله پنجم – تنظیمات سرور های Worker

برای اضافه کردن گره worker یک دستور را باید اجرا کنید ، که شامل جزئیات مورد نیاز کلاستر از قبیل آدرس IP و پورت API سرور master می باشد و یک توکن امنیتی می باشد. فقط سروری که توکن را قبول می کند می تواند به کلاستر متصل شود.

به مسیر فضای کاری Workspace بروید و یک playbook به نام workers.yml ایجاد کنید :

```
nano ~/kube-cluster/workers.yml
```

متن زیر را در فایل قرار دهید :

```
-hosts: master
 become: yes
 gather_facts: false
 tasks:
- name: get join command
```

```
    shell: kubeadm token create --print-join-command
    register: join_command_raw
  - name: set join command
    set_fact:
      join_command: "{{ join_command_raw.stdout_lines [۰]}}"
-hosts: workers
 become: yes
 tasks:
  - name: join cluster
    shell: "{{ hostvars['master'].join_command }} >> node_joined.txt"
    args:
      chdir: $HOME
      creates: node_joined.txt
```

توضیحات این playbook :

- در ابتدا دستور اتصال اجرا می شود که باید در سرور worker وارد شود و قالب دستور به این صورت است : kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hash sha256:<hash>. این دستور با قرار دادن مقادیر token و hash تکمیل می شود.
- قسمت دوم فقط دستور اتصال به گره ها یا سرور های worker را اجرا می کند که پس از تکمیل شدن این دستور هر دو سرور ، قسمتی از کلاستر خواهند بود.

فایل را ذخیره کرده و خارج شوید.

فایل playbook را اجرا کنید :

```
ansible-playbook -i hosts ~/kube-cluster/workers.yml
```

خروجی این اجرا مشابه زیر است :

```
Output
PLAY [master]****

TASK [get join command]****

changed: [master]

TASK [set join command]*****
```

```
ok: [master]
PLAY [workers]*****
TASK [Gathering Facts]*****
ok: [worker1]
ok: [worker2]
TASK [join cluster]*****
changed: [worker1]
changed: [worker2]
PLAY RECAP*****
master                     : ok=2 changed=1 unreachable=0    failed=0
worker1                    : ok=2 changed=1 unreachable=0    failed=0
worker2                    : ok=2 changed=1 unreachable=0    failed=0
```

با اضافه شدن سرور های worker کلاستر شما به صورت کامل تنظیم شده و آماده بهره برداری است و worker ها آماده اجرای بارهای کاری ارسالی هستند. قبل از زمان بندی برنامه ، صحت عملکرد کلاستر را بررسی می کنیم.

## مرحله ششم – بررسی کلاستر

به ممکن است به دلایلی مانند از دسترس خارج شدن یک سرور worker یا خرابی ارتباط بین سرور های master و worker ، راه اندازی کلاستر با مشکل مواجه شود. در این قسمت بررسی می کنیم که کلاستر و سرور ها به درستی کار کنند.

در ابتدا وضعیت گره ها یا سرور های متصل به سرور master را در سرور master را بررسی می کنیم. ابتدا به سرور master متصل می شویم :

```
ssh ubuntu@master_ip
```

سپس دستور زیر را برای بررسی وضعیت سرور اجرا کنید :

```
kubectl get nodes
```

خروجی باید مشابه زیر باشد :

```
Output
NAME      STATUS ROLES     AGE VERSION
master    Ready master    1d v1.10.1
worker1   Ready <none>    1d v1.10.1
worker2   Ready <none>    1d v1.10.1
```

اگر وضعیت STATUS سرور ها در حالت ready بود ، به این معنی است که این قسمت از سرور به درستی کار می کند و آماده انجام کارها می باشند.

و اگر هر وضعیت STATUS هر سروری در حالت NotReady باشد به این معنی است که تنظیمات آن سرور هنوز تمام نشده است. حدود ۵ تا ۱۰ دقیقه قبل از اجرای مجدد دستور kubectl get node صبر کنید ، سپس مجددا خروجی را بررسی کنید. اگر هنوز وضعیت سرور ها NotReady بود ، باید دستورات مراحل قبل را مجددا بررسی و اجرا کنید.

پس از اطمینان از صحت عملکرد کلاستر ، برنامه Nginx را به عنوان یک نمونه زمان بندی یا Schedule می کنیم.

## مرحله هفتم – اجرای یک برنامه در کلاستر

اکنون می توانید هر برنامه ای که نیاز به مخزن دارد را اجرا کنید. برای نمونه برنامه Nginx را بر روی کلاستر اجرا می کنیم. از دستوراتی که در ادامه آورده می شوند می توانید برای اجرای هر برنامه دیگری استفاده کنید.

در سرور master دستور زیر را برای ایجاد یک deployment از nginx وارد کنید :

```
kubectl run nginx --image=nginx --port 80
```

deployment یک نوع از شئ Kubernetes می باشد که وجود همیشگی تعدادی pod در حال اجرا را در یک قالب تعریف شده تضمین می کند ، حتی اگر در زمان اجرای کلاستر pod از دسترس خارج شود. قبل از deployment یک pod با یک مخزن از داکر موجود در Ngingx Docker Image ایجاد می شود.

سپس برای ایجاد یک سرویس با نام nginx برای در اختیار عموم قرار دادن این برنامه ، دستور زیر را وارد کنید. این کار از طریق NodePort انجام می شود ، به این صورت که یک پورت دلخواه را برای اتصال به یک pod در یک سرور دیگر باز می کند :

```
kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
```

بقیه سرویس هایی که از انواع شئ های Kubernetes هستند به صورت داخلی فعالیت می کنند و برای سرویس دهی به کاربران محلی و خارجی هستند. همچنین این سرویس ها می توانند عملیات تقسیم بار را در درخواست ها انجام دهند و جزو جدایی ناپذیر Kubernetes هستند که به صورت سلسله مراتبی با اجزای دیگر فعالیت می کند.

دستور زیر را اجرا کنید :

```
kubectl get services
```

خروجی این دستور مانند زیر است :

```
Output
NAME         TYPE CLUSTER-IP      EXTERNAL-IP PORT(S)       AGE
kubernetes   ClusterIP 10.96.0.1       <none> 443/TCP        1d
nginx        NodePort    10.109.228.209   <none>     80:nginx_port/TCP   40m
```

همانطور که در خروجی مشاهده می کنید Nginx بر روی پورت ۸۰ کار می کند. Kubernetes یک پورت تصادفی بالاتر از ۳۰۰۰۰ که مطمئن است به سرویس دیگری تعلق نگرفته است را به این برنامه می دهد.

برای بررسی صحت عملکرد آدرس http://worker_1_ip:nginx_port یا http://worker_2_ip:nginx_port را در مرورگر سرور محلی خود وارد کنید. در این قسمت باید صفحه Nginx را مشاهده کنید.

برای حذف برنامه Nginx ابتدا سرویس Nginx را از سرور master حذف کنید :

```
kubectl delete service nginx
```

دستور زیر را وارد کنید تا از حذف شدن برنامه مطمئن شوید :

```
kubectl get services
```

خروجی باید مشابه زیر باشد :

```
Output
NAME         TYPE CLUSTER-IP      EXTERNAL-IP PORT(S)      AGE
kubernetes   ClusterIP 10.96.0.1        <none> 443/TCP       1d
```

سپس deployment را حذف کنید :

```
kubectl get deployments
Output
No resources found.
```

## پایان

در این آموزش شما توانستید یک کلاستر مخازن Kubernetes در Ubuntu 16.04 بوسیله Kubeadm راه اندازی کنید ، و در مراحل بعدی باید بتوانید برنامه ها و سرویس ها بیشتری را در کلاستر deploy کنید. در ادامه چند لینک برای راهنمایی و آموزش بیشتر شما قرار داده ایم :

Dockerizing applications - لیستی از مثال هایی برای آموزش استفاده از مخازن داکر.

Pod Overview - توضیحات بیشتر درباره pod ها و ارتباط آن ها با اجزا دیگر Kubernetes. این pod ها در همه جای Kubernetes هستند و فهم بهتر آن ها به شما کمک زیادی می کند.

Deployments Overview - مرور بر مفهوم deployment است. خیلی مهم است که بدانید deployments ها چطور کارها را کنترل می کنند.

Services Overview - سرویس های دیگری را که توسط اجزا Kubernetes استفاده می شوند را بررسی می کند. درک انواع سرویس ها و گزینه هایی که برای اجرای هر دو برنامه های stateless و statefull هستند ، ضروری است.

مفاهیم مهم دیگر Volumes, Ingresses و Secrets. هستند که برای deploy برنامه ها در Kubernetes بسیار مفید هستند. Kubernetes ، امکانات و ویژگی های زیادی دارد که در مستندات رسمی Kubernetes بهترین منبع برای مطالعه می باشد.