# باسمه تعالی

# مقدمه

این مستند به منظور توضیح نحوه پیاده سازی و استفاده از تکنولوژی ها و زیرساخت های آماده شده در پروژه پایلوت سرویس ابری هوش مصنوعی نگاشته شده است. در این مستند تلاش شده است توضیحات فنی به نحوی آماده شود که خواننده بتواند به تنهایی نمونه مشابه ای از این زیرساخت را برای خود بالا بیاورد.

# استفاده از الگوریتم های یادگیری ماشین

کد های ارائه شده دو فایل مختلف دارند که یکی مخصوص الگوریتم Logistic Regression و دیگری مخصوص الگوریتم Random Forest است.

با دستور زیر میتوانید یک دیتاست را به عنوان داده ورودی به مدل بدهید و مدل یادگرفته شده را خروجی بگیرید:

Python <model-type.py> -T <train-file.csv> -M <model-output-name>

همچنین با دستور زیر می توانید مدل یادگرفته شده را بر روی داده های جدید اعمال کنید و نتیجه را مشاهده کنید.

Python <model-type.py> -P <test-file.csv> -M <model-name> -O <output.csv>

# نحوه کانفیگ کردن و نصب Docker و Kubernetes

از سیستم عامل لینوکس (ubuntu 20.04) با نسخه کرنل ۵.۸.۰ استفاده شده است.

**نصب درایور انویدیا:**

* sudo apt install nvidia-driver-450
* sudo reboot

[**نصب داکر**](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository)**:**

* sudo apt-get update
* sudo apt-get install \  
  apt-transport-https \  
  ca-certificates \  
  curl \   
  gnupg-agent \  
  Software-properties-common
* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
* sudo add-apt-repository \  
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
  $(lsb\_release -cs) \  
  stable"
* sudo apt-get update
* sudo apt-get install docker-ce docker-ce-cli containerd.io

[**نصب Nvidia Container Toolkit**](https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#setting-up-nvidia-container-toolkit)**:**

* distribution=$(. /etc/os-release;echo $ID$VERSION\_ID) \  
  && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \  
  && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
* sudo apt-get update
* sudo apt-get install -y nvidia-docker2
* sudo systemctl restart docker

**تنظیمات داکر:**

sudo nano /etc/docker/daemon.json

محتویات زیر را در فایل بالا زخیره کنید:

{

    "default-runtime": "nvidia",

    "runtimes": {

        "nvidia": {

            "path": "nvidia-container-runtime",

            "runtimeArgs": []

        }

    }

}

برای بالا آوردن یک کلاستر کوبرنتیز (kubernetes) روی یک سرور میتوان از راه‌های مختلفی استفاده کرد. اما برای کلاستر کردن چندین سرور میتوان از ابزار kubeadm استفاده کرد. در این فرایند، یک نود به عنوان master و سایر نود ها worker خواهند بود.

در ابتدا پکیج‌های kubeadm، kubelet، و kubectl نصب میشوند. مراحل نصب در این [لینک](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/) آمده است. وظایف هر کدام از این پکیج‌ها در زیر آمده است. در این پروژه از نسخه‌ی ۱.۱۴.۱۰ این پکیج‌ها استفاده شده است.

* پکیج kubeadm: از این ابزار برای ساخت یک کلاستر کوبرنتیز استفاده میشود.
* پکیج kubectl: این ابزار برای اجرای دستورات روی یک کلاستر کوبرنتیز می باشد.
* پکیج kubelet: این پکیج وظیفه‌ی نگه‌داری و سرپا نگه داشتن هسته‌ی یک کلاستر کوبرنتیز را بر عهده دارد.

مراحل نصب به شرح زیر است

* sudo apt-get update && sudo apt-get install -y apt-transport-https curl
* curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
* cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
* deb https://apt.kubernetes.io/ kubernetes-xenial main
* EOF
* sudo apt-get update
* sudo apt-get install -y kubelet=1.14.10-00 kubeadm=1.14.10-00 kubectl=1.14.10-00
* sudo apt-mark hold kubelet kubeadm kubectl

**غیرفعال کردن swap**

sudo swapoff –a

پس از نصب پکیج‌های مورد نیاز میتوان یک کلاستر روی یک یا چند نود ایجاد کرد. این کار با دستور kubeadm init انجام میشود. جزئیات این فرایند در این [لینک](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/) آمده است.

**ایجاد کلاستر**

sudo kubeadm init --apiserver-advertise-address=<master-ip> --pod-network-cidr=10.244.0.0/16

**نصب flannel**

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

از بین بردن taint روی نود مستر در کلاستر

kubectl taint nodes --all node-role.kubernetes.io/master-

نصب داشبورد کوبرنتیز

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml>

پس از نصب داشبورد با استفاده از دستورات این [لینک](https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md) یک توکن ساخته میشود تا بتوان به داشبورد کوبرنتیز دسترسی پیدا کرد.

**نصب** [**Nvida k8s device plugin**](https://github.com/NVIDIA/k8s-device-plugin#enabling-gpu-support-in-kubernetes)**:**

kubectl create -f <https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.8.0/nvidia-device-plugin.yml>

**نصب [kubeflow](https://v1-0-branch.kubeflow.org/docs/started/k8s/kfctl-k8s-istio/):**

نصب [volume provisioner](https://github.com/rancher/local-path-provisioner#deployment):

kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml

پیش‌فرض سازی کلاس ذخیره‌سازی local-path که در دستور قبلی نصب شد.

kubectl patch storageclass local-path -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

نصب [kfctl](https://v1-0-branch.kubeflow.org/docs/started/k8s/kfctl-k8s-istio/" \l "provisioning-of-persistent-volumes-in-kubernetes):

* wget <https://github.com/kubeflow/kfctl/releases/download/v1.0.2/kfctl_v1.0.2-0-ga476281_linux.tar.gz>
* tar -xvf kfctl\_v1.0.2-0-ga476281\_linux.tar.gz
* sudo install ./kfctl /usr/bin
* export KF\_NAME=kubeflow-v1.0
* export BASE\_DIR=<path to a base directory>
* export KF\_DIR=${BASE\_DIR}/${KF\_NAME}
* export CONFIG\_URI="<https://raw.githubusercontent.com/kubeflow/manifests/v1.0-branch/kfdef/kfctl_k8s_istio.v1.0.2.yaml>"
* mkdir -p ${KF\_DIR}
* cd ${KF\_DIR}
* kfctl apply -V -f ${CONFIG\_URI

# استفاده از الگوریتم Image classification

توضیحات کامل نحوه اجرا و استفاده از این الگوریتم به تفصیل در [این لینک](https://github.com/newsha1998/DML-Lab/blob/master/Documentation/DistributedData-ParallelTrainingwithKubeflowPytorch.md) آماده شده است.

# راه اندازی و اجرای SPARK بر روی خوشه Kubernetes

توضیحات کامل و مراحل اجرای یک جاب SPARK بر روی یک خوشه Kubernetes به تفصیل در [این لینک](https://github.com/newsha1998/DML-Lab/blob/master/Documentation/RunPysparkOnK8s.md) آماده شده است.

# نحوه استفاده از HDFS در میان کدهای SPARK و PyTorch

توضیحات کامل نحوه استفاده از API های موجود HDFS برای ذخیره سازی و خواندن داده ها در کدهای Spark و PyTorch به تفصیل در [این لینک](https://github.com/newsha1998/DML-Lab/blob/master/Documentation/UsingHDFS.md) آماده شده است.