PNDA Helm Deployment on EKS based K8S Cluster

Cloud Native PNDA development guide on AWS EKS:

This document describes how you can setup Cloud Native PNDA on AWS

Prerequisites:

- AWS account
- Windows OS / Oracle VM Virtual Box Manager with latest Ubuntu OS based VM
- VM needs to have following installed
    1. AWS Cli
    2. Docker
    3. Kubernetes (kubectl)
    4. Eksctl
    5. Helm Client

AWS CLI Installation and Setup:

Use following link for AWS CLI setup

https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html

Once setup is done then configure aws cli as per below,

$ **aws configure**

AWS Access Key ID [None]: *Enter your access key ID*

AWS Secret Access Key [None]: *Enter your secret access key*

Default region name [None]: *Enter your region*

Default output format [None]: *json*

Docker Installation:

Follow the steps as mentioned in an official documentation to install Docker,

https://docs.docker.com/install/linux/docker-ce/ubuntu/#set-up-the-repository

Kubernetes(kubectl) Installation:

In your terminal run the following commands:

- curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt) /bin/linux/amd64/kubectl
- chmod +x ./kubectl
- sudo mv ./kubectl /usr/local/bin/kubectl

EKSCTL Installation:

Run the following commands in your terminal

- curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
- sudo mv /tmp/eksctl /usr/local/bin
- eksctl version

Helm client 2.14 Installation:

$curl -L https://git.io/get_helm.sh | bash -s -- --version v2.14.3

$helm version

EKS CLUSTER CREATION:

Step 1) Create EKS cluster on AWS:

You can create EKS cluster on AWS by using either CLI or AWS GUI

Creating EKS cluster through CLI by step a or b:

1. **EKS cluster creation command from cli:**

```
sudo eksctl create cluster --name pnda \
 --region us-east-2 \
--version 1.14 \
```

```
--nodegroup-name pnda-nodes \
--node-type t3.xlarge \
--nodes 5 \
--nodes-min 2 \
--nodes-max 6 \
–managed
```

2. Using Config files: create pnda_eks_cluster.yaml file

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: pnda
  region: es-east-1
nodeGroups:
  - name: ng-pnda-1
    instanceType: t3.large
    desiredCapacity: 10
    volumeSize: 80
    ssh:
      allow: true # will use ~/.ssh/id_rsa.pub as the default ssh key
  - name: ng-pnda-2
    instanceType: t3.xlarge
    desiredCapacity: 2
    volumeSize: 100
    ssh:
      publicKeyPath: ~/.ssh/ec2_id_rsa.pub
```

```
$ eksctl create cluster -f pnda_eks_cluster.yaml
```

Note: If you needed to use an existing VPC, you can use config file with VPC details,

```
vpc:
  subnets:
    private:
      eu-north-1a: { id: subnet-0ff156e0c4a6d300c }
      eu-north-1b: { id: subnet-0549cdab573695c03 }
      eu-north-1c: { id: subnet-0426fb4a607393184 }
```

Creating EKS cluster thru GUI:

Follow the steps as mentioned in following link,

https://docs.bitnami.com/aws/get-started-eks/

Note: Creating EKS cluster thru GUI is time taking and harder as compared to using CLI.

Step 2) Setup alb Load balancer for created cluster.

(https://docs.aws.amazon.com/eks/latest/userguide/alb-ingress.html)

Run the following commands to setup load balancer,

- eksctl utils associate-iam-oidc-provider --region us-east-2 --cluster pnda --approve
- curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/master/docs/examples/iam-policy.json
- aws iam create-policy --policy-name ALBIngressControllerIAMPolicy --policy-document file://iam-policy.json
- Attach ingress policies for Node Instance role.

    IAM Console --> Roles --> search for the NodeInstanceRole.

    (Example: eksctl-pnda-eks-NodeInstanceRole-xxxxxx.) --> Attach policy select ingressController-iam-policy.

- kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.4/docs/examples/rbac-role.yaml
- curl -sS "https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.4/docs/examples/alb-ingress-controller.yaml" > alb-ingress-controller.yaml
- Open alb-ingress-controller and edit ingress-class, cluster-name, aws-vpc-id and region.

kubectl edit deployment.apps/alb-ingress-controller -n kube-system

```
spec:
  containers:
  - args:
    - --ingress-class=alb
      --cluster-name=pnda
      --aws-vpc-id=vpc-05b0819933d5440cb
      --aws-region=us-east-2
```

- kubectl apply -f alb-ingress-controller.yaml
- kubectl get pods -n kube-system à This should return running alb ingress controller pod.

Step 3) Helm tiller Installation on EKS cluster:

$helm init

Create rbac-config.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: tiller
    namespace: kube-system
```

$kubectl create -f rbac-config.yaml

$helm init --service-account tiller --upgrade

$helm version

Note: Default EBS General Purpose SSD(gp2) volume types only supports read-write-once persistent volume (PV) access-mode. The PV which we create for Deployment Manager requires read-write-many access-mode, as multiple pods (Deployment Manager, Spark Operatoretc..) share same PV. Hence, we have to create/setup EFS CSI driver which supports read-write-many access-mode for a PV.

Step 4) To deploy Amazon EFS CSI driver to an Amazon EKS cluster

1. kubectl apply -k "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/?ref=master"
2. aws eks describe-cluster --name pnda --query "cluster.resourcesVpcConfig.vpcId" --output text
    a. (Above command will return VPC ID and use the same in below command)
3. aws ec2 describe-vpcs --vpc-ids **vpc-exampledb76d3e813** --query "Vpcs[].CidrBlock" --output text (Above command returns VPC CODR range and use the same while adding rule for NFS inbound traffic)
    a. Create a security group that allows inbound NFS traffic for your Amazon EFS mount points.
    b. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/
    c. Choose Security Groups in the left navigation pane, and then Create security group.
    d. Enter a name and description for your security group, and choose the VPC that your Amazon EKS cluster is using.
    e. Choose Create and then Close to finish.
4. Add a rule to your security group to allow inbound NFS traffic from your VPC CIDR range.
    a. Choose the security group that you created in the previous step.
    b. Choose the Inbound Rules tab and then choose Edit rules.
    c. Choose Add Rule, fill out the following fields, and then choose Save rules.
        i. Type: NFS
        ii. Source: Custom. Paste the VPC CIDR range.
        iii. Description: Add a description, such as "Allows inbound NFS traffic from within the VPC."

Step 5) Create AWS EFS

(https://us-east-2.console.aws.amazon.com/efs/home?region=us-east-2#/get-started)

Note: Use same region as EKS cluster created region in above URL

There are 2 ways to create AWS elastic file system

1. Using GUI:
    a. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/ .
    b. Choose File systems in the left navigation pane, and then choose Create file system.
    c. On the Create file system page, choose Customize.
    d. On the File system settings page, you don't need to enter or select any information, but can if desired, and then select Next.
    e. On the Network access page, for Virtual Private Cloud (VPC), choose your VPC.**Note:** If you don't see your VPC, at the top right of the console, make sure that the region that your VPC is in is selected.
    f. Under Mount targets, if a default security group is already listed, select the X in the top right corner of the box with the default security group name to remove it from each mount point, select the security group that you created in a previous step for each mount target, and then select Next.
    g. On the File system policy page, select Next.
    h. On the Review and create page, select Create.

(Or)

2. Using CLI:

Follow the steps from following link,

https://docs.aws.amazon.com/efs/latest/ug/wt1-create-efs-resources.html

Step 6) Setup K8S-EFS provisioner

1. Download k8s-efs repository from git,
    a. git clone https://github.com/kubernetes-incubator/external-storage
2. Switch to the deploy directory
    a. cd external-storage/aws/efs/deploy/
3. Apply rbac permissions
    a. Kubectl apply -f rbac.yaml
4. Modify manifest.yaml. In the configmap section change the system.id: and aws.region: to match the details of the EFS you created. Change dns.name if you want to mount by your own DNS name and not by AWS's *file-system-id*.efs.*aws-region*.amazonaws.com. See following attachment for manifest.yaml,

5. Apply the manifest

   kubectl apply -f manifest.yaml

6. Check PV and PVC created properly

   kubectl get pv, pvc (should return efs volume with aws-efs storage class)

Step 7) Setup Console/Kafka/Grafana/etc.. service types as load balancer for Helm charts

Console: ~/pnda/pnda-helm-chart/cloud-pnda/values.yaml

```
consoleFrontend:
  image:
    repository:  pnda/console-frontend
    tag: 2.1.2
    pullPolicy: IfNotPresent
  service:
    type: LoadBalancer
    port: 80
```

Kafka: ~/pnda/pnda-helm-chart/cloud-pnda/charts/kafka-manager

```
service:
  type: LoadBalancer
  port: 9000
  annotations: {}
```

Grafana: ~/pnda/pnda-helm-chart/cloud-pnda/charts/grafana/values.yaml

```
service:
  type: LoadBalancer
  port: 80
  targetPort: 3000
    # targetPort: 4181 To be used with a proxy extraContainer
  annotations: {}
  labels: {}
```

Step 8) **Add storage class as gp2 for all other pvc and aws-efs for deployment manager pvc.**

storageClass: aws-efs

storageClass: gp2

Step 9) Install PNDA on eks with helm

```
pnda@pnda:~/pnda-helm-chart$ helm install --name pnda --namespace pnda cloud-pnda
```

(helm install --name pnda --namespace pnda cloud-pnda)

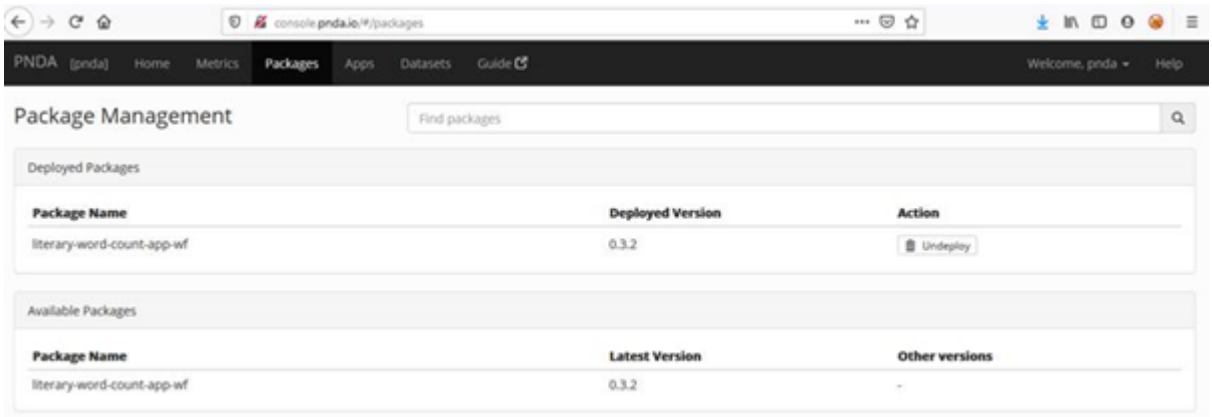
Step 10) Setup DNS alias for external IP's in Route 53 to access the DNS like console.pnda.io within private network

Step 11) Access the console front end from any one of the ec2 instance.

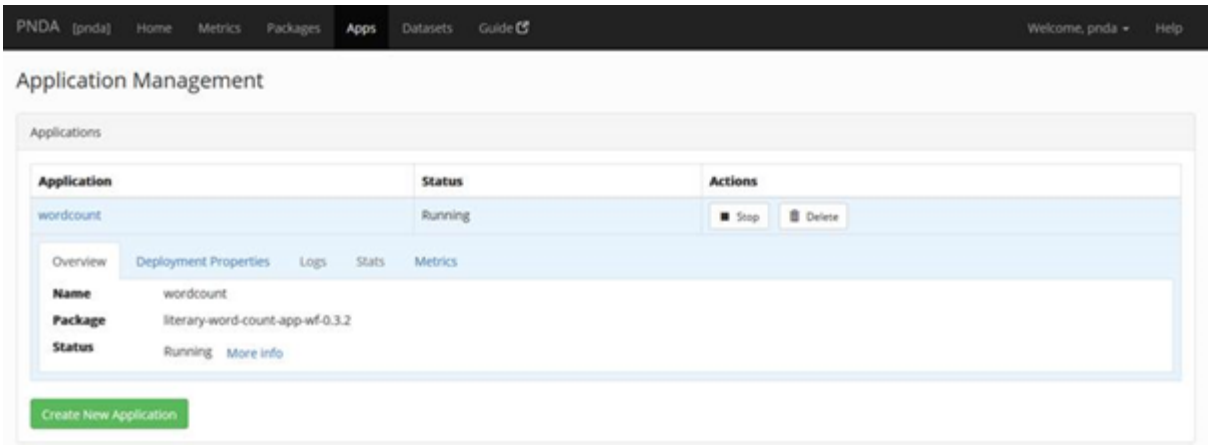
PNDA UI: http://console.pnda.io/

Step 12) Testing the deployment of SparkStreaming

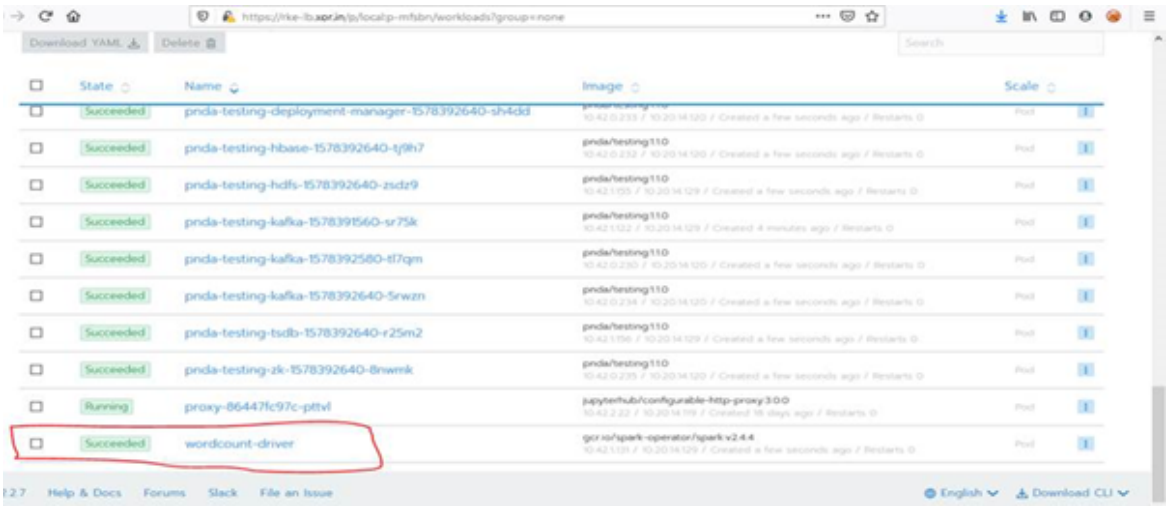
1. To test the deployment of SparkStreaming we have created an example app literary-word-count-app-wf-0.3.2.tar.gz.
2. To upload the package to the cloud-pnda platform:
    a. kubectl -n pnda port-forward service/pnda-package-repository 8888
3. It is possible to temporarily expose the package repository API with kubectl port forwarding:
    a. curl -XPUT "http://localhost:8888/packages/literary-word-count-app- wf-0.3.2.tar.gz?user.name=" --data-binary "@literary-word-count-app- wf-0.3.2.tar.gz"
4. You can see the uploaded package details under available packages in Package Management and then deploy the package by clicking Deploy button.



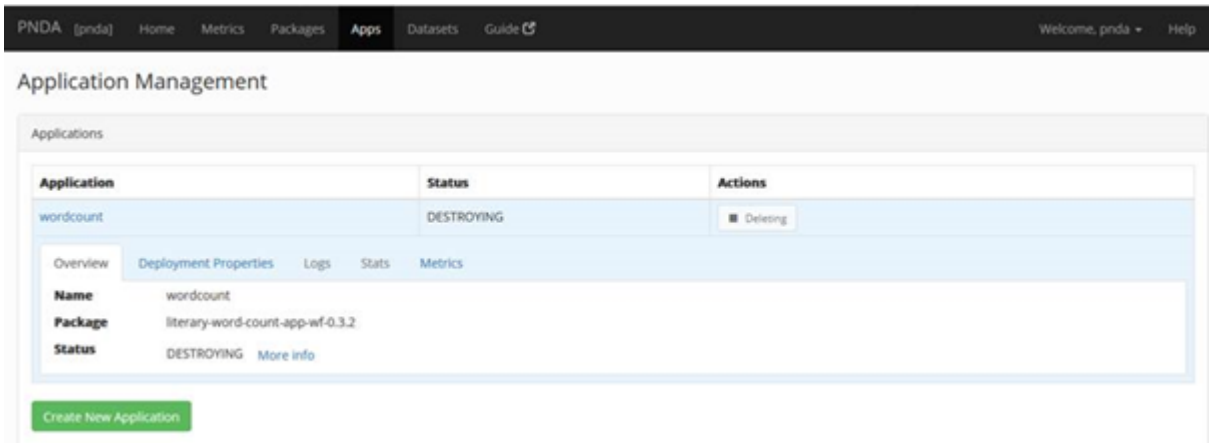
5. After successfully deploying the package, create/install the application in the Apps Tab



6. Verify the application on K8S cluster(Tested on Rancher cluster).



7. To delete the application, you have to click on delete button of particular application under Apps Tab,



Sample Spark Streaming example is in following Git repo,

https://github.com/sreenivasa-xor/SparkStreaming-Example-Application.git

Step 13) Clean-up

Delete PNDA on eks cluster

% helm del –purge pnda

Delete EKS Cluster:

% eksctl delete cluster --region=us-east-2 --name=pnda