Labs Workbench on AWS

Features

- Auto-Scaling
- CloudWatch Monitoring
- Free-Tier
- Marketplace for Amazon Machine Images and Services

Free Tier

Micro instances are eligible for the AWS free usage tier. For the first 12 months following your AWS sign-up date, you get up to 750 hours of micro instances each month. When your free usage tier expires or if your usage exceeds the free tier restrictions, you pay standard, pay-as-you-go service rates.

Learn more about free usage tier eligibility and restrictions.
The free tier has the following monthly limits:

- Up to 750 compute hours
- Up to 30GB of General Purpose SSD volume storage
- Up to 5GB of S3 storage
- Up to 750 hours of Amazon RDS database usage
- 1M requests to Amazon Lambda
- 1GB of SPICE capacity on Amazon QuickSight

Manual Instance Deployment

Steps should seem fairly familiar:

1. Sign into AWS Console, and choose Services Compute EC2
2. Choose Instances on the left side
3. Click "Launch instance"
4. On the left, choose Marketplace
5. Search for, and select, CoreOS
6. Choose t2.micro (free) for your instance size
7. Configure instance details (networking / monitoring / auto-scaling / protection from accidental termination / etc)
8. Configure tags (completely optional and arbitrary)
9. Configure security groups (you'll likely want to add at the very least SSH / HTTP / HTTPS)
10. Review and click "Launch" and you'll be prompted to assign or create a key pair for this instance

I was then able to SSH in and clone ndslabs-startup to try to run our cluster - startup went fairly smoothly (aside from my free instance being entirely too small).

I needed to assign a DNS rule pointing www.awsdev.ndslabs.org to this instance

Now, I am running into the same issues as my VirtualBox VM:

503 5.0.0 Need RCPT (recipient)

Gotchas

- Restarting an instance assigns it a new public IP (probably unless we pay more for a static IP)

Potential Problems

1. API server goes into CrashLoopBackoff (can't connect to etcd or Kubernetes API, despite seemingly correct security groups)
    - Possibly addressed by **NDS-774** - Getting issue details... STATUS
    - Need to try with templates from https://github.com/craig-willis/ndslabs-startup/tree/azure/templates
2. Gluster, and how we do things with storage in general (these could possibly be coerced to run on compute nodes)
    - Likely addressed by **NDS-775** - Getting issue details... STATUS
3. Each compute node gets a public IP, so these could potentially run the loadbalancer pod(s) if we open port 80 on all of them
    - This is likely not a problem, although I did not think to verify this until after my cluster was already down

Prerequisites (for all)

Steps:

1. Create an AWS account
2. Create an IAM user / group with programmatic (API) level access and the following permissions:
    a. AmazonEC2FullAccess
    b. AmazonS3FullAccess
    c. AmazonRoute53FullAccess
    d. AmazonVPCFullAccess
    e. IAMFullAccess
3. Download and install **wget**, **python** and **pip**
    a. Run the following commands:

```
pip install awscli
aws configure
```

    b. You will be prompted for your Access Key ID, Secret Access Key, Default Region, and Default Output Format (text, table, etc)

Deploying via kube-up.sh

See: https://kubernetes.io/docs/getting-started-guides/aws/#kube-up-bash-script

More details: https://medium.com/@canthefason/kube-up-i-know-what-you-did-on-aws-93e728d3f56a#.hue9bb1yn

kube-up.sh will produce a test cluster (i.e. not production-ready) for evaluation on AWS.

NOTE: Apparently, CoreOS is now longer supported

```
Michaels-MacBook-Pro-2:~ lambert8$ export KUBERNETES_PROVIDER=aws; wget -q -O - https://get.k8s.io | bash
Creating a kubernetes on aws...
coreos is no longer supported by kube-up; please use jessie instead
Michaels-MacBook-Pro-2:~ lambert8$
```

Setup

1. Follow the Prerequisite steps above
    a. Then, set some defaults using environment variables:

```
export MASTER_SIZE=t2.micro
export NODE_SIZE=t2.micro
export KUBE_OS_DISTRIBUTION=jessie
export KUBERNETES_PROVIDER=aws; wget -q -O - https://get.k8s.io | bash
```

    b. This will perform the following actions using your AWS account:
        i. Upload installation files to S3
        ii. Create IAM roles
        iii. Create a key pair and publish to AWS
        iv. Create VPC
        v. Create Subnet

vi. Create Internet Gateway
vii. Create Routing Table
viii. Create Security Groups
ix. Create and attach persistent volume to master
x. Create master instance
xi. Create node instances

2. You should see output similar to the following:

```
Michaels-MacBook-Pro-2:~ lambert8$ export KUBERNETES_PROVIDER=aws; wget -q -O - https://get.k8s.io | bash
'kubernetes' directory already exist. Should we skip download step and start to create cluster based on
it? [Y]/n
Skipping download step.
Creating a kubernetes on aws...
... Starting cluster in us-west-2a using provider aws
... calling verify-prereqs
... calling verify-kube-binaries
... calling kube-up
Starting cluster using os distro: jessie
Uploading to Amazon S3
+++ Staging server tars to S3 Storage: kubernetes-staging-4736c124943095293b0c8e5bcf3aba1e/devel
upload: ../../../var/folders/2n/nylhqdm96dxcqpjhjpkw04dw0000gn/T/kubernetes.XXXXXX.qjwmPifr/s3/bootstrap-
script to s3://kubernetes-staging-4736c124943095293b0c8e5bcf3aba1e/devel/bootstrap-script
Uploaded server tars:
  SERVER_BINARY_TAR_URL: https://s3.amazonaws.com/kubernetes-staging-4736c124943095293b0c8e5bcf3aba1e
/devel/kubernetes-server-linux-amd64.tar.gz
  SALT_TAR_URL: https://s3.amazonaws.com/kubernetes-staging-4736c124943095293b0c8e5bcf3aba1e/devel
/kubernetes-salt.tar.gz
  BOOTSTRAP_SCRIPT_URL: https://s3.amazonaws.com/kubernetes-staging-4736c124943095293b0c8e5bcf3aba1e
/devel/bootstrap-script
Creating master IAM profile: kubernetes-master-kubernetes-kubernetes-vpc
Creating IAM role: kubernetes-master-kubernetes-kubernetes-vpc
Creating IAM role-policy: kubernetes-master-kubernetes-kubernetes-vpc
Creating IAM instance-policy: kubernetes-master-kubernetes-kubernetes-vpc
Adding IAM role to instance-policy: kubernetes-master-kubernetes-kubernetes-vpc
Creating minion IAM profile: kubernetes-minion-kubernetes-kubernetes-vpc
Creating IAM role: kubernetes-minion-kubernetes-kubernetes-vpc
Creating IAM role-policy: kubernetes-minion-kubernetes-kubernetes-vpc
Creating IAM instance-policy: kubernetes-minion-kubernetes-kubernetes-vpc
Adding IAM role to instance-policy: kubernetes-minion-kubernetes-kubernetes-vpc
Generating public/private rsa key pair.
Your identification has been saved in /Users/lambert8/.ssh/kube_aws_rsa.
Your public key has been saved in /Users/lambert8/.ssh/kube_aws_rsa.pub.
The key fingerprint is:
SHA256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX lambert8@Michaels-MacBook-Pro-2.local
The key's randomart image is:
Using SSH key with (AWS) fingerprint: SHA256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Creating vpc.
Adding tag to vpc-3b71085c: Name=kubernetes-vpc
Adding tag to vpc-3b71085c: KubernetesCluster=kubernetes
Using VPC vpc-3b71085c
Adding tag to dopt-0600d361: Name=kubernetes-dhcp-option-set
Adding tag to dopt-0600d361: KubernetesCluster=kubernetes
Using DHCP option set dopt-0600d361
Creating subnet.
Adding tag to subnet-96b51ddf: KubernetesCluster=kubernetes
Using subnet subnet-96b51ddf
Creating Internet Gateway.
Using Internet Gateway igw-346b6f50
Associating route table.
Creating route table
Adding tag to rtb-8835d0ee: KubernetesCluster=kubernetes
Associating route table rtb-8835d0ee to subnet subnet-96b51ddf
Adding route to route table rtb-8835d0ee
Using Route Table rtb-8835d0ee
Creating master security group.
Creating security group kubernetes-master-kubernetes.
Adding tag to sg-e71a229f: KubernetesCluster=kubernetes
Creating minion security group.
Creating security group kubernetes-minion-kubernetes.
Adding tag to sg-d51a22ad: KubernetesCluster=kubernetes
Using master security group: kubernetes-master-kubernetes sg-e71a229f
```

```
Using minion security group: kubernetes-minion-kubernetes sg-d51a22ad
Creating master disk: size 20GB, type gp2
Adding tag to vol-0b6d9ca24a37b5ac7: Name=kubernetes-master-pd
Adding tag to vol-0b6d9ca24a37b5ac7: KubernetesCluster=kubernetes
Allocated Elastic IP for master: PUB.LIC.IP.ADR
Adding tag to vol-0b6d9ca24a37b5ac7: kubernetes.io/master-ip=PUB.LIC.IP.ADR
Generating certs for alternate-names: IP:PUB.LIC.IP.ADR,IP:172.20.0.9,IP:10.0.0.1,DNS:kubernetes,DNS:
kubernetes.default,DNS:kubernetes.default.svc,DNS:kubernetes.default.svc.cluster.local,DNS:kubernetes-
master
Starting Master
Adding tag to i-04147c9d36d22c618: Name=kubernetes-master
Adding tag to i-04147c9d36d22c618: Role=kubernetes-master
Adding tag to i-04147c9d36d22c618: KubernetesCluster=kubernetes
Waiting for master to be ready
Attempt 1 to check for master nodeWaiting for instance i-04147c9d36d22c618 to be running (currently
pending)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be running (currently pending)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be running (currently pending)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be running (currently pending)
Sleeping for 3 seconds...
 [master running]
Attaching IP PUB.LIC.IP.ADR to instance i-04147c9d36d22c618
Attaching persistent data volume (vol-0b6d9ca24a37b5ac7) to master
2017-03-13T22:49:32.744Z        /dev/sdb        i-04147c9d36d22c618        attaching        vol-
0b6d9ca24a37b5ac7
Cluster "aws_kubernetes" set.
User "aws_kubernetes" set.
Context "aws_kubernetes" set.
Switched to context "aws_kubernetes".
User "aws_kubernetes-basic-auth" set.
Wrote config for aws_kubernetes to /Users/lambert8/.kube/config
Creating minion configuration
Creating autoscaling group
 0 minions started; waiting
 0 minions started; waiting
 0 minions started; waiting
 0 minions started; waiting
 4 minions started; ready
Waiting for cluster initialization.

  This will continually check to see if the API for kubernetes is reachable.
  This might loop forever if there was some uncaught error during start
  up.

.....................................................................................................
.....................Kubernetes cluster created.
Sanity checking cluster...
Attempt 1 to check Docker on node @ PUB.LIC.IP.ADR ...working
Attempt 1 to check Docker on node @ PUB.LIC.IP.ADR ...working
Attempt 1 to check Docker on node @ PUB.LIC.IP.ADR ...working
Attempt 1 to check Docker on node @ PUB.LIC.IP.ADR ...working

Kubernetes cluster is running.  The master is running at:

  https://PUB.LIC.IP.ADR

The user name and password to use is located in /Users/lambert8/.kube/config.

... calling validate-cluster
Waiting for 4 ready nodes. 0 ready nodes, 4 registered. Retrying.
Waiting for 4 ready nodes. 2 ready nodes, 4 registered. Retrying.
Found 4 node(s).
NAME                                         STATUS    AGE
ip-172-20-0-154.us-west-2.compute.internal   Ready     46s
ip-172-20-0-155.us-west-2.compute.internal   Ready     37s
ip-172-20-0-207.us-west-2.compute.internal   Ready     48s
ip-172-20-0-228.us-west-2.compute.internal   Ready     50s
Validate output:
```

```
NAME                 STATUS    MESSAGE              ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health": "true"}
etcd-1               Healthy   {"health": "true"}
Cluster validation succeeded
Done, listing cluster services:

Kubernetes master is running at https://PUB.LIC.IP.ADR
Elasticsearch is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services
/elasticsearch-logging
Heapster is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services/heapster
Kibana is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services/kibana-logging
KubeDNS is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services/kube-dns
kubernetes-dashboard is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services
/kubernetes-dashboard
Grafana is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services/monitoring-
grafana
InfluxDB is running at https://PUB.LIC.IP.ADR/api/v1/proxy/namespaces/kube-system/services/monitoring-
influxdb

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

Kubernetes binaries at /Users/lambert8/kubernetes/cluster/
You may want to add this directory to your PATH in $HOME/.profile
Installation successful!
```

3. Add the output of the kube-up.sh script to your path, and access your cluster
    a. You may want to make an alias from kubectl.sh to kubectl, if the minor difference bothers you (like myself)

```
Michaels-MacBook-Pro-2:~ lambert8$ export PATH=$PATH:/Users/lambert8/kubernetes/cluster/
Michaels-MacBook-Pro-2:~ lambert8$ kubectl.sh get nodes
NAME                                         STATUS     AGE
ip-172-20-0-154.us-west-2.compute.internal   Ready      2m
ip-172-20-0-155.us-west-2.compute.internal   Ready      2m
ip-172-20-0-207.us-west-2.compute.internal   Ready      2m
ip-172-20-0-228.us-west-2.compute.internal   Ready      2m
Michaels-MacBook-Pro-2:~ lambert8$ kubectl.sh get pods
No resources found.
Michaels-MacBook-Pro-2:~ lambert8$
```

4. Start NDS Labs Workbench services
    a. Clone the ndslabs-startup repo
    b. Edit the source to change all **kubectl** calls in ndslabs-up.sh to **kubectl.sh**
    c. Run ./ndslabs-up.sh

Teardown

To shut down your test cluster:

```
Michaels-MacBook-Pro-2:~ lambert8$ kube-down.sh
Bringing down cluster using provider: aws
Deleting instances in VPC: vpc-3b71085c
Deleting auto-scaling group: kubernetes-minion-group-us-west-2a
Deleting auto-scaling launch configuration: kubernetes-minion-group-us-west-2a
Deleting auto-scaling group: kubernetes-minion-group-us-west-2a
Deleting auto-scaling group: kubernetes-minion-group-us-west-2a
Deleting auto-scaling group: kubernetes-minion-group-us-west-2a
Waiting for instances to be deleted
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
Waiting for instance i-04147c9d36d22c618 to be terminated (currently shutting-down)
Sleeping for 3 seconds...
All instances deleted
Releasing Elastic IP: PUB.LIC.IP.ADR
Deleting volume vol-0b6d9ca24a37b5ac7
Cleaning up resources in VPC: vpc-3b71085c
Cleaning up security group: sg-d51a22ad
Cleaning up security group: sg-e71a229f
Deleting security group: sg-d51a22ad
Deleting security group: sg-e71a229f
Deleting VPC: vpc-3b71085c
Deleting DHCP option set: dopt-0600d361
Deleting IAM Instance profiles
Removing role from instance profile: kubernetes-master-kubernetes-kubernetes-vpc
Deleting IAM Instance-Profile: kubernetes-master-kubernetes-kubernetes-vpc
Delete IAM role policy: kubernetes-master-kubernetes-kubernetes-vpc
Deleting IAM Role: kubernetes-master-kubernetes-kubernetes-vpc
Removing role from instance profile: kubernetes-minion-kubernetes-kubernetes-vpc
Deleting IAM Instance-Profile: kubernetes-minion-kubernetes-kubernetes-vpc
Delete IAM role policy: kubernetes-minion-kubernetes-kubernetes-vpc
Deleting IAM Role: kubernetes-minion-kubernetes-kubernetes-vpc
Done
```

Deploying a Production Cluster via kops

See: https://kubernetes.io/docs/getting-started-guides/aws/#supported-production-grade-tools-with-high-availability-options

More details: https://github.com/kubernetes/kops

More detailed instructions for AWS: https://github.com/kubernetes/kops/blob/master/docs/aws.md

Features:

- No CoreOS support
- Automate the provisioning of Kubernetes clusters in (AWS) "other platforms planned"
- Deploy Highly Available (HA) Kubernetes Masters
- Supports upgrading from kube-up
- Built on a state-sync model for dry-runs and automatic idempotency
- Ability to generate Terraform configuration
- Supports custom add-ons for `kubectl`
- Command line autocompletion
- Community supported!

Installation

1. Follow the Prerequisite steps above
2. Download kubectl binary:

```
# OS X
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.
com/kubernetes-release/release/stable.txt)/bin/darwin/amd64/kubectl

# Linux
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.
com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

# Windows
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.
com/kubernetes-release/release/stable.txt)/bin/windows/amd64/kubectl.exe
```

3. Add binary to PATH:

```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```

4. Download / install kops:

```
# OS X
brew update && brew install kops

# Linux
export KOPS_VERSION=1.5.3
curl https://github.com/kubernetes/kops/releases/download/${KOPS_VERSION}/kops-linux-amd64 -o kope-linux-
amd64
chmod +x kops-linux-amd64                # Add execution permissions
mv kops-linux-amd64 /usr/local/bin/kops   # Move the kops to /usr/local/bin
```

Usage

1. Set up some cluster parameters:

```
export NAME=awstest.ndslabs.
org

export KOPS_STATE_STORE=s3://nivenly-state-
store

kops create cluster --zones us-west-2a $NAME
```

2.

Deploying a Production Cluster via kube-aws (CoreOS-only)

See: https://coreos.com/kubernetes/docs/latest/kubernetes-on-aws.html

More details: https://github.com/coreos/kube-aws

Deploying via Kargo

See: https://kubernetes.io/docs/getting-started-guides/kargo/

More details: https://github.com/kubernetes-incubator/kargo

Some random blog post running through the steps: http://mntdevops.com/2016/11/15/caas-1/