**CONTINOUS DEPLOYMENTS USING SPINAKKER ON AWS AND KUBERNETES**

**Spinnaker –** part of delivery pipeline; will take care of deployment phase

* Can be used with **Continuous Integration**
* Can deploy to all major **cloud** providers
* Opensource **multi-cloud** CD platform
* Started in 2014 by Netflix
* **Reliable deployments; high velocity; fast and open**
* **Complements k8s** by supporting powerful deployment

**INSTALLING SPINNAKER**

**Vagrant (on local machine)** – HashiCorp

* **Download Vbox**
* **vagrant up** on that Vagrant file UNDER ung cinlone from git:
* **git clone** <https://github.com/in4it/spinnaker-course>
* **copy the URL after installation then paste it to the browser to connect to Spinnaker**

*USEFUL COMMANDS:*

**vagrant halt** stop vagrant machine

**vagrant destroy** erase v-m

**vagrant box list** base boxes on ur v-m

* **SPINNAKER CONCEPTS:**
* **CD platform; releases ur software**
* Machines will be grouped by **cluster**

***2 Core sets:***

* ***Cluster Management*** *view and manage ur cluster resources in the cloud*
* ***Deployment Management*** *create and manage continuous delivery workflow; describe the way app should be deployed*

**Account** contains acc name and a credential that is able to authenticate access and secret access key (AWS/GOOGLE/OpenStack/Azure)

**Instance** can be a VM instance on AWS or a k8s pod

**Server group** collection of instances, running the deployed software

**Cluster** spinnaker deployment logical grouping of server grps

* **PROVIDERS:**
* Set of virtual resources that spinnaker has control over
* Can have multiple accounts per provider
* Supported prov: google app engine, AWS, Azure, DC/OS, Google Compute Engine, K8s. OpenStack
* **DEPLOYMENT STRATEGIES**
* Determine how the new app version will be rolled out
* Red/Black or Green/Blue **starts the same** amount of instance in a new server grp
* Rolling Red/Black or Green/Blue **per instance or per group** of instances
* Canary **analysis;** replace small amount of instances; test whether the new version doesn’t have any issues
* **PIPELINES**
* Once the cluster is set up u can start this
* Part of deployment management
* Can be triggered manually, automatically

**DEPLOY WITH SPINNAKER ON AWS**

* **AWS CONCEPTS**
* **VPC** isolated network of AWS cloud; private
* **EC2 IAM Role** controlusers/app access of using specific amazon api
* **EC2 Key pair** pub-key cryptography to encrypt and decrypt login info
* **AMI** image with the needed info to start a virtual machine
* **AWS S3** block storage, uses buckets
* **Access Keys** IDs and Secret Access keys; use to sign programmatic requests

**DISTINCTION** between **managed** and **managing** acc:

**Managing** acc used to log into AWS; only one acc

**Managed** acc where spinnaker actually is able to modify resources; can have multiple acc

**DEMO:**

***CREATE VPC WITH SUBNET AND CREATE IAM ROLE***

VPC > CREATE FIRST VPC WITH SUBNET THERE

SUBNETS > CREATE ANOTHER SUBNET > NAME:subnet2 > deploy on Oregon > CIDR: 10.0.1.0/24 > CREATE

IAM > ROLES > CREATE ROLE > SELECT EC2 > NEXT > ROLENAME: BASEIAMROLE > CREATE

POLICIES > CREAT EPOLICY > REPLACE THE MANAGING ACC ID BY PASTING THE ID FROM: SUPPORT DROP DOWN > **ACC NUMBER** FROM **THAT SUPPORT CENTER** PAGE > **PASTE** IT THERE IN BOTH MANAGING AND MANAGED ACC ID

|  |
| --- |
| { |
|  |

|  |
| --- |
| "Version": "2012-10-17", |
|  |

|  |
| --- |
| "Statement": [{ |
|  |

|  |
| --- |
| "Action": "sts:AssumeRole", |
|  |

|  |
| --- |
| "Resource": [ |
|  |

|  |
| --- |
| "arn:aws:iam***::${MANAGING\_ACCOUNT\_ID}***:role/spinnakerManaged", |
|  |

|  |
| --- |
| "arn:aws:iam***::${MANAGED\_ACCOUNT\_ID}:***role/spinnakerManaged" |
|  |

|  |
| --- |
| ], |
|  |

|  |
| --- |
| "Effect": "Allow" |
|  |

|  |
| --- |
| }] |
|  |

}

GIVE IT A NAME > CREATE

***CREATE USERS/ROLES AND ATTACH SOME POLICIES***

USERS > ADD A USER > NAME:SPINNAKER > PROGRAMMATIC ACCESS > NEXT > ATTACH EXISTING POLICY: PowerUserAccess > CREATE > COPY THOSE KEYS AND USER ARN

POLICIES > CREATE POLICIES > JSON > Spinnaker Policy Role in git hub or ung resources don >

|  |
| --- |
| Policy: |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| "Version": "2012-10-17", |
|  |

|  |
| --- |
| "Statement": [{ |
|  |

|  |
| --- |
| "Effect": "Allow", |
|  |

|  |
| --- |
| "Action": [ "ec2:\*" ], |
|  |

|  |
| --- |
| "Resource": "\*" |
|  |

|  |
| --- |
| }, |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| "Effect": "Allow", |
|  |

|  |
| --- |
| "Action": "iam:PassRole", |
|  |

|  |
| --- |
| "Resource": "arn:aws:iam:***:${MANAGING\_ACCOUNT\_ID}:***role/BaseIAMRole" |
|  |

|  |
| --- |
| }] |
|  |

|  |
| --- |
| } |
|  |

NEXT > ROLENAME: SpinnakerPassRole > CREATE

ROLE > CREATE > EC2 > PowerUserAcess > Attach permission: SpinnakerPassRole > NEXT > ROLENAME: SpinnakerManaged > CREATE ROLE

SELECT THE ROLE SpinnakerManaged > Trust Realtionships TAB > EDIT TRUST > POLICY DOCUMENT FOR GIT OR CLONE U HAVE: EDIT THE ARN THERE THE ONE U’VE COPIED EARLIER

|  |
| --- |
| { |
|  |

|  |
| --- |
| "Version": "2012-10-17", |
|  |

|  |
| --- |
| "Statement": [{ |
|  |

|  |
| --- |
| "Sid": "1", |
|  |

|  |
| --- |
| "Effect": "Allow", |
|  |

|  |
| --- |
| "Principal": { |
|  |

|  |
| --- |
| "AWS": ***"${AUTH\_ARN}"*** |
|  |

|  |
| --- |
| }, |
|  |

|  |
| --- |
| "Action": "sts:AssumeRole" |
|  |

|  |
| --- |
| }] |
|  |

}

NOW UPDATE THE POLICY

***CREATE EC2 KEYPAIR***

EC2 > KEYPAIRS > CREATE > NAME:Spinnakercourse (in .pem file)

***CREATE S3 BUCKET***

S3 > CREATE > NAME: spinnaker-debian-repo-shien > NEXT > NEXT > NEXT > CREATE BUCKET

***CREATE ACCESS KEYS***

THE ONE U GENERATED EARLIER OR JUST CREATE A NEW UNDER THAT USER: Spinnaker

***ADDING ACCOUNT TO SPINNAKER; CONFIGURING THIS ACCOUNT***

IN CLI/TERMNAL:

SSH ON THAT MACHINE OR JUST USE THE PEM FILE OR PUTTY

THEN GO TO SPINNKER UI, BEFORE U CAN RESIZE IT U MUST SHUTDOWN THE MACHINE OF THAT INSTANCE: **shutdown -h no; U CAN ALSO TURN ON THE MACHINE**

under that spinnaker-course dir > scripts > restart the spinnaker; just run the command there

Then > hal config storage s3 edit - -access-key-id <***uraccessidhere>***  - -secret-access-key

It will not prompt u for ur secret access key > ENTER

IT WIL HAVE AN ERROR

**vi ~./hal/config**

**SEARCH FOR /S3**

**DELETE THERE the entire line that has “endpoint” > SAVE IT**

**NOW RUN: ./5-deploy-spinnaker.sh**

**THEN RESTART IT: ./6-restart-spinnaker.sh**

**AWS > S3 > U CAN SEE THAT THERE IS A NEW BUCKET CREATED**

**IN CLI/TERMINAL:**

**hal config storage s3 edit - -access-key-id <*uraccessidhere>*  - -secret-access-key**

It will not prompt u for ur secret access key > ENTER

**hal config provider aws account add my-aws-account - -account-id <uraccountid> - -assume-role role/spinnakerManaged**

**hal config provider aws account edit my-aws-account - -region us-east-1**

**hal config provider aws enable**

**then AGAIN deploy the spinnaker then RESTART IT AGAIN**

* **IMMUTABLE INFRASTRUCTURE**
* Consists of immutable components that are replaced on every deployment **rather than being updated**
* **DEPLOYMENT PIPELINE USING AWS**

**DEMO:**

**IN TERMINAL:**

**GO TO THE ARTIFACTS DIR OF THE spinnaker-course of git repo**

**INSTALL AWS CLI**

**aws configure - -profile default;** fill in the info need there such as keys, region and output

**install unzip**

**unzip artifact.zip**

**go that directory artifact**

**aws s3 sync . s3://spinnaker-debian-repo-shien** *----that’s the path of s3 u created; that bucket name>*

AWS>

u can now check ur S3 bucket; there are now some pkgs there

CLICK: UNDER POOL DIR > node-demo-app….pkg > MAKE PUBLIC BUTTON > U CAN SEE SUCCESS

BACK TO ROOT DIR OF THE BUCKET IN AWS > dists dir > xenial > main > click binary > MORE DROPDOWN : MAKE PUBLIC > MAKE PUBLICJ

AWS > EC2 > SECURITY GROUPS > CREATE > NAME: demo-app-ec2 > select default vpc u created > INBOUND > ADD RULE: CUSTOM TCP > PORT TANGE: 3000 > SOURCE: 10.0.0.0/8 > CREATE

CREATE NEW SECGRP:

AWS > EC2 > SECURITY GROUPS > CREATE > NAME: demo-app-alb > select default vpc u created > ADD RULE: HTTP > PORT RANGE: 80 > SOURCE: ANYWHERE > CREATE

IN TERMINAL:

**cp /opt/rosco/config/rosco.yml .hal/default/profiles**

**vi .hal/default/profiles/rosco.yml**

**edit:**

**configDir = /opt/rosco/config/packer/**

**under the #chocolateyRepo…**

**debianRepository:** [**https://s3.amazonaws.com/spinnaker-debian-repo-shien/**](https://s3.amazonaws.com/spinnaker-debian-repo-shien/) **xenial main**

**in aws: line**

**enabled: true**

**add this then SAVE IT**

* **baseImage:**

**id: ubuntu xenial**

**shortDescription: v16.04**

**detailDescription: Ubuntu Xenial 16.04 LTS**

**packageType: deb**

**GO TO spinnaker-course dir > scripts dir > ./5-deploy-spinnaker.sh**

**Then restart it again ./6-restart-spinnaker.sh**

**sytemctl status rosco.service**

**IN SPINNAKER UI:**

**ACTIONS DROP DOWN > NEW APP > NAME: demoapp > email > CREATE APP**

**LOADBALANCER > CREATE > ALB > NAME: my-aws-acc > sec group alb u created earlier > TARGET GROUPS > PORT 3000 > PATH / > CREATE**

**PIPELINES > demo-app > CREATE > ADD STAGE > STAGE NME TYPE DROP DOWN: BAKE > PKG NAME > node-demo-app > SAVE**

**ADD ANOTHER STAGE > TYPE: DEPLOY > DEPENDS ON: BAKE > ADD SEVER GROUP > STRATEGY: HIGHLANDER > TARGET GROUPS > demoapp-targetgroup > SEC GRP > demo-app-ec2 > MICRO UTILITY > NANO > ADVANCED SETTINGS: KEY NAME Spinnakercourse < IAM Instance Profile: BaseIAMRole > ADD > SAVE**

**START MANUAL EXECUTION > VIEW BAKERY DETAILS > COPY THE AMI there after the successful install > GO TO AWS AMI PASTE IT: SEARCH THE ONE U COPIED > TO VERIFY:**

**EC2 > Packer Builder > LOAD BALANCERS >U CAN SEE THE THE DNS NAME THERE U CAN COPY AND PASTE IT INTO THE BROSWSER**

**SPINNAKER WITH JENKINS**

**ON AWS > LAUNCH INSTANCE > UBUNTU**

**BUILDING PIPELINES**

* executes one or more stages with a specific stage type
* every stage except one, has **dependency** on one of the previous stage
* run stages in **parallel**

***STAGE TYPES:***

* ***Bake*** *bakes an image uses rosco (spinnaker) and packer (Hashicorp); creates AMI in specific region that install a pkg*
* ***Check Precondition*** *checks before continuing – don’t continue if cluster size is smaller than x (variable:expression)*
* ***Clone Server Group*** *makes a copy of a server grp with higher capacity (red/black strategy)*
* ***Deploy*** *creates new server group; deploy after baking a new image*
* ***Destroy Server Group*** *destroys a previous/newest/oldest server grp; deploy new version of the app*
* ***Disable Cluster*** *disabling your staging environment*
* ***Modify Scaling Process*** *parameters; disable health-check of a server group*
* ***Pipeline*** *runs pipeline*
* ***Resize Server Group*** *scale out from 5 instances to 10*
* ***Rollback cluster*** *executes rollback on a cluster and applies to all server groups running in the cluster*
* ***Scale Down/ Shrink Cluster*** *removes server grp; only keep the newest grp*
* ***Script*** *runs a custom python script*
* ***Tag Image*** *with key value-pair*
* ***Wait*** *waits for the next stage*
* ***Webhook*** *invokes a webhook based on URL, using a JSON payload*

**DEMO:**