UTSAV BHETWAL

385-479-2425 |Salt Lake City, UT 84121| utsav_bhetwal2008@yahoo.com

www.linkedin.com/in/utsav-bhetwal https://github.com/festivo1 https://hub.docker.com/u/festivo1

Devops Documentation

Courses Covered:

1. Git and Github
2. Jenkins
3. IBMurbancode deploy
4. Ansible
5. Terraform
6. Docker
7. Kubernetes
8. Linux file system and bash scripting

1. Git and Github Account(https://github.com/festivo1)
   To install Git
   - sudo yum install git

   To configure git

   - git config - - global user.name <username>
   - git config - - global user.email <email_addr>
   - git config - - system core.editor vim
     Alternatively you may store system-wide configuration values in the file directly:
     - /etc/gitconfig which corresponds to the --system
     - ~/.gitconfig or ~/.config/git/config which corresponds to the --global
     - .git/config in a repository which corresponds to –local
     - Note: Files lower in the list override files higher in the list

   <u>Creating a repository and adding content</u>
   - git init <repo> to create or initialize a git repo in specified dir
   - git add <filename> used for tracking the file in the repo
   - git commit -m"some messages" → commit message
   - git push -u origin master → push to master branch
   - git checkout -b devEnv → create devEnv branch and goes to that branch
   - git rm <filename> → stop tracking the file
   - git clone user@server:<original_repo_path><local_repo_path>
   - git log → gives info about commits and changes within the repo

eg.

```
git log --oneline → gives the oneline log of each commit
abcb6b8 (HEAD -> master, origin/master, origin/HEAD)  changes code
eec89e6 terraform_installation added
de99e88 Merge branch 'master' of https://github.com/festivo1/devops_tools
7c1d333 tomcat_installation
87e3637 Update bash_for_jenkins.sh
f7cbb00 Update ansible_installation.txt
b6f5e7a added the installation commands for jenkins, ansible, ibmucd
e210dfb Update ruby_for_jenkins.rb
efb235a Update python_for_jenkins.py
b3e32bc Update bash_for_jenkins.sh
2628c3f java installlation command added
3197abe some comments in the file
00943c6 added readme.md file
f590f70 Update bash_for_jenkins.sh
e1c1e5a first commit
```

- git log -p → gives detail logs
- git log -- <filename> → log of a particular file
- git log -- oneline <filename>
- git log –graph--decorate

Merging and pushing updates:

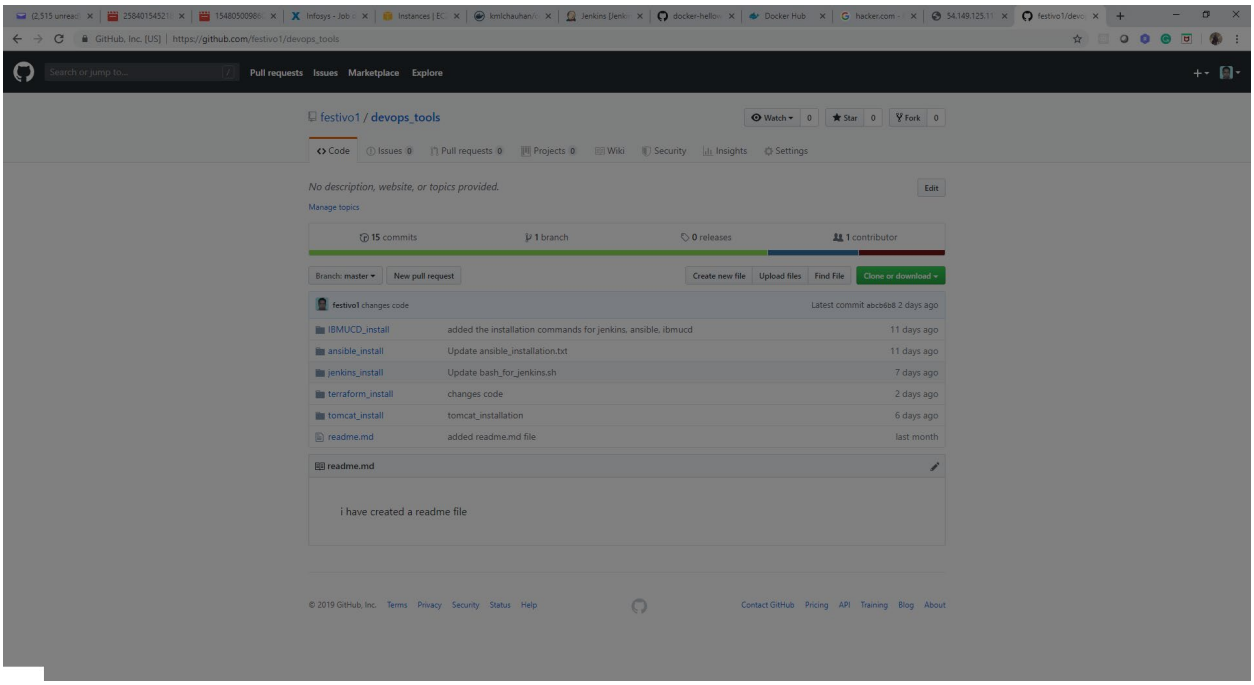
Branches may be pushed to remote sources like master

- git push origin - -all

merging brings branches together(must be in the branch you are merging into)

- git merge <target branch>
  Note: merging branches where files may have changed in diverging ways in called a merge conflict. You must resolve this conflict yourself

  GithubAccount: https://github.com/festivo1/devops_tools



2. JENKINS And CI

Continuous integration (CI) is a software development practice where members of a team integrate their work frequently, at least daily, leading to multiple integrations per day. Continuous Delivery (CD) is a software development discipline where software is built so that it can be released to production at any time.

ci→ automated testing→ continuous deployment to production

Jobs are runnable tasks that are controlled or monitored by Jenkins.

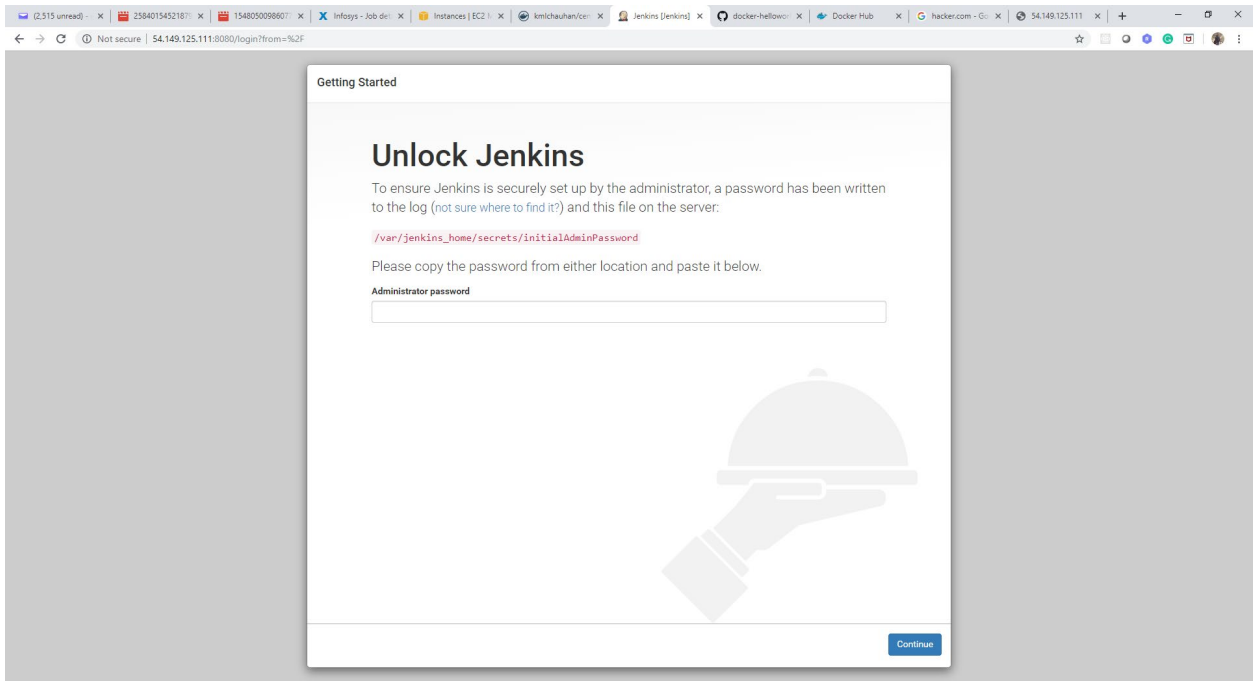
Types of jobs:Freestyle, Monitor external, Matrix (Multiconfiguration), and Maven

A build step is a single task within a project.

A trigger is a criteria for starting a new pipeline run or build.

An artifact is an immutable file that is generated during a build or pipeline run.

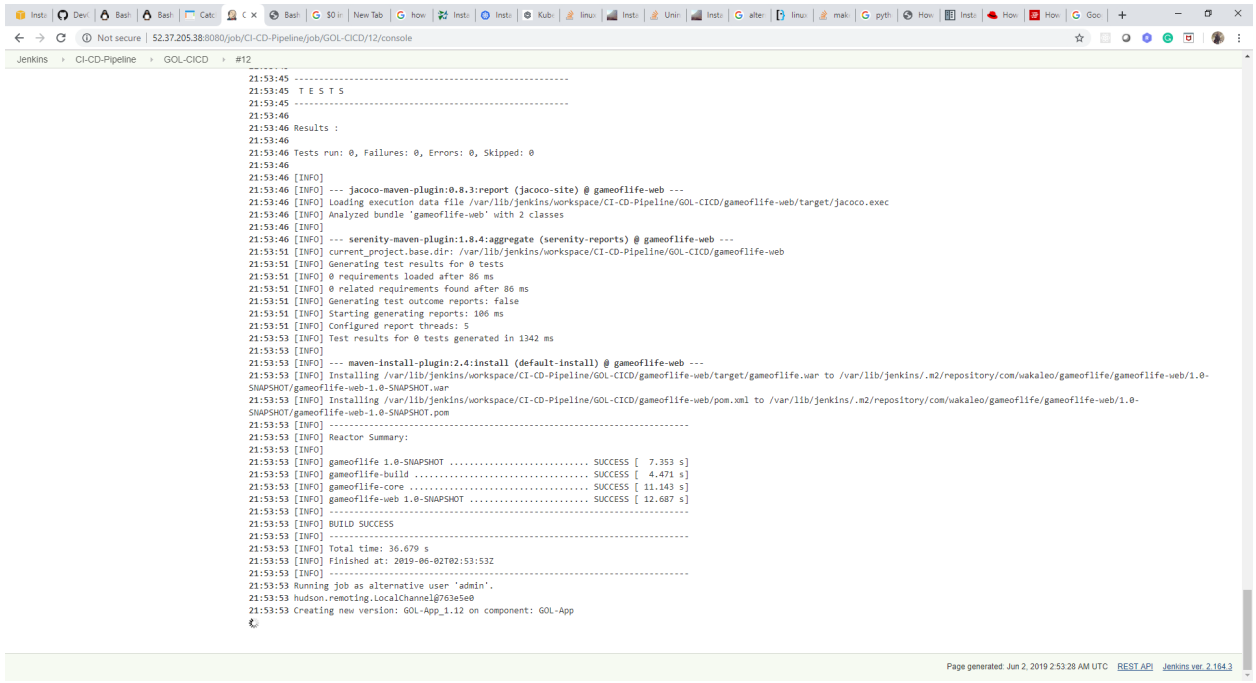
A repository is a location that holds items that need to be retrieved, source code, artifacts, jenkinsfiles etc.



3. IBM urbanCode Deploy(CD)

IBM urbancode deploy plugins needs to be downloaded and It should be uploaded as a plugin in Jenkins. Create a ucd-server(8443) instance and run it. You get a ucd-agent zip files from the server. Unzip it in ucd agent and create an agent. Then in agent you install any web server like httpd to show static content of your application.

- Bluegreen strategy applied



```
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorial-HelloWorldWi
thAnt.html
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorial-tasks-filese
ts-properties.html
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorial-tasks-filese
ts-properties.zip
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorial-writing-task
s-src.zip
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorial-writing-task
s.html
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/tutorials.html
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/using.html
  inflating: ucd-agent-install/opt/apache-ant-1.8.4/manual/usinglist.html
  inflating: ucd-agent-install/opt/groovy-1.8.8/ANTLR-LICENSE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/ASM-LICENSE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/CLI-LICENSE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/JSR223-LICENSE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/LICENSE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/NOTICE.txt
  inflating: ucd-agent-install/opt/groovy-1.8.8/conf/groovy-starter.conf
  inflating: ucd-agent-install/opt/groovy-1.8.8/embeddable/groovy-all-1.8.8.jar

  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/ant-1.8.3.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/ant-antlr-1.8.3.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/ant-junit-1.8.3.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/ant-launcher-1.8.3.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/antlr-2.7.7.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/asm-3.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/asm-analysis-3.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/asm-commons-3.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/asm-tree-3.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/asm-util-3.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/bsf-2.4.0.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/commons-cli-1.2.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/commons-logging-1.1.1.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/gpars-1.0-beta-3.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/groovy-1.8.8.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/hamcrest-core-1.1.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/ivy-2.2.0.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/jansi-1.6.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/jline-1.0.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/jsp-api-2.0.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/jsr166y-1.7.0.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/junit-4.10.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/servlet-api-2.4.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/xmlpull-1.1.3.1.jar
  inflating: ucd-agent-install/opt/groovy-1.8.8/lib/xstream-1.4.1.jar
  inflating: ucd-agent-install/zostoolkit.zip
[root@ip-172-31-50-61 opt]# ls
apache-tomcat-8.5.41          tomcat          ucd-agent.zip
apache-tomcat-8.5.41.tar.gz  ucd-agent-install
[root@ip-172-31-50-61 opt]# cd ucd-agent-install/
[root@ip-172-31-50-61 ucd-agent-install]# ls
binary.zip                     install-agent.sh
config.zip                     install-many-agents.bat
convertFilesForZos.sh          install-many-agents.sh
example.agent.install.properties  install.properties
groovy                         install.with.groovy.xml
ibmi-compatability-fix.sh      lib
install                        monitor.version
install-agent.bat              opt
install-agent-from-file.bat    README.txt
install-agent-from-file.sh     zostoolkit.zip
[root@ip-172-31-50-61 ucd-agent-install]# sudo yum install java-1.8.0-openjdk-de
vel -y
Last metadata expiration check: 1:07:30 ago on Tue 04 Jun 2019 04:15:44 PM UTC.
Package java-1.8.0-openjdk-devel-1:1.8.0.212.b04-1.el8_0.x86_64 is already insta
lled.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-50-61 ucd-agent-install]#
```

Fig: ucd_agent

4. Ansible

- A configuration management tool(CMT) for infrastructure of a code.
- Helps in provisioning the instances

STEPS:

1. install ansible
2. configure host file
[controlmachine]
control ansible_connection=local

[webserver]
define public ip's of instance

3. define httpd.yaml
4. import devops_demo.pem file in tmp dir of ansible server from your local machine
rsync devops_demo.pem ubuntu@public_ip:/tmp

ansible: to install multiple packages ginga format
#play playbook to install httpd server
ansible-playbook -i httpd.yaml --private-key=~/devops_demo.pem -u ec2-user

5.#create roles:
ansible-galaxy init httpd
6.copy the created httpd.yaml in roles/httpd/tasks/main.yml

```
---
# tasks file for httpd
        - name: install httpd server
          yum:
             name: httpd
             state: present

        - name: start httpd server
          service:
                name: httpd
                state: started
        - name: copy index.html file
          copy:
               src: /home/ubuntu/index.html
               dest: /var/www/html/index.html
```

important concept:

In enterprise level: we don't use host file present in default location(/etc/ansible/hosts)

but what we can do is placing the hosts/inventory file in the repository.

eg. redhat satellite server and ipam installation:

spins the hosts using some configuration and ipam server saves all the ips specifically for all the

environment; dhcp server searches the vacant ip's then go to ipam server create a records and entry there

give a specific host name to it and again go back to dhcp server, in dhcpd.conf configure the ip hostname and mac-address

Then go to redhat server after giving some information we spin up the host machine. once host machine is up and running

ansible play will execute on that machine.

so let's not use the etc/ansible/hosts: so just copy the etc/ansible/hosts to aws-inventory

:cp /etc/ansible/hosts aws-inventory // aws-inventory file is created

remove the configuration in etc/ansible/hosts

mv index.html /roles/files/

Note: as we removed the configuration from /etc/ansible/hosts, and copied to aws-inventory, we have to now specify

that inventory file as well as httpd.yaml playbook

ansible-playbook -i aws-inventory --private-key=~/devops_demo.pem
httpd.yaml -u ec2-user
error:
change the source file only to index.html // no absolute path

ansible SEcond lecture

installing java8 in ubuntu machine using roles
ansible-galaxy init java8
create some tasks inside roles/java8/tasks/main.yml
create one aws_ec2.yaml file for configuration using the tag instead of
putting ip-address

for this tag configuration:
add the tag parameter in the specific instances

and to connect to aws from your terminal
first: install python pip
#sudo apt-get install python-pip -y
second: install aws-cli using pip
#pip install awscli
Note: create user ansible
IAM--> users--> add user--. provide username and check to
programmatic access and click next permissions
--> add user to the already created group--> next tags--> next review-->
create user
after creating your user download.csv file where there is access_key and
secret_access_key for logging into aws
#aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-west-2
Default output format [None]: json

#aws iam get-user //show the user configuration with username
and userId
for aws ec2 plugin:
Also need to install boto3: sudo apt-get install python-boto3
check with: ansible-inventory -i aws_ec2.yaml --list
Plugin name and the yml file name must end with the plugin name
like aws_ec2 or demo.aws_ec2 and so on.
the aws_ec2.yml file looks like this

```
---
plugin: aws_ec2
regions:
    - us-west-2
strict_permissions: False
strict: false
hostnames:
    - tag:Name // NO space between tag and Name if space error occurs
    - private-ip-address
filters:
    tag:sub_env: test
keyed_groups:
    - key: tags.ansible_groups
      separator: ''
    - key: tags.ansible_group
      separator: ''
    - key: tags.Name[:5]
      separator: ''
    - key: tags
      prefix: tag
    - key: placement.region
      prefix: aws_region
    - prefix: instance_type
      key: instance_type
compose:
    working_environment: tags.working_environment
    ansible_host: private_ip_address
```

```
ubuntu@ip-172-31-61-64:~$
ubuntu@ip-172-31-61-64:~$
ubuntu@ip-172-31-61-64:~$ ansible-inventory -i aws_ec2.yml --graph
@all:
  |--@Ansib:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@aws_ec2:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@aws_region_us_west_2:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@db:
  |  |--AnsibleManagedDB
  |--@instance_type_t2_micro:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@loadbalancer:
  |  |--AnsibleManagedLoadBalancer
  |--@tag_Name_AnsibleManagedDB:
  |  |--AnsibleManagedDB
  |--@tag_Name_AnsibleManagedLoadBalancer:
  |  |--AnsibleManagedLoadBalancer
  |--@tag_Name_AnsibleWebserver2:
  |  |--AnsibleWebserver2
  |--@tag_Name_AnsibleWebservre1:
  |  |--AnsibleWebservre1
  |--@tag_Type_AnsibleManagedEC2Instances:
  |  |--AnsibleWebservre1
  |--@tag_Type_AnsibleManagedEC2Instances:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |--@tag_ansible_group_db:
  |  |--AnsibleManagedDB
  |--@tag_ansible_group_loadbalancer:
  |  |--AnsibleManagedLoadBalancer
  |--@tag_ansible_group_webserver:
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@tag_sub_env_test:
  |  |--AnsibleManagedDB
  |  |--AnsibleManagedLoadBalancer
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
  |--@ungrouped:
  |--@webserver:
  |  |--AnsibleWebserver2
  |  |--AnsibleWebservre1
ubuntu@ip-172-31-61-64:~$
```

```
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1039-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  7 04:14:54 UTC 2019

  System load:  0.0               Processes:           89
  Usage of /:   32.0% of 7.69GB   Users logged in:     0
  Memory usage: 20%               IP address for eth0: 172.31.61.64
  Swap usage:   0%

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with

     snap install microk8s --classic

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

2 packages can be updated.
2 updates are security updates.


*** System restart required ***
Last login: Thu Jun  6 22:33:36 2019 from 67.177.1.62
ubuntu@ip-172-31-61-64:~$ ansible-playbook -i aws_ec2.yml --private-key=devops_demo.pem site.yml -u ec2-user
ERROR! the playbook: site.yml could not be found
ubuntu@ip-172-31-61-64:~$ cd ansible-pythonapp-deployment
ubuntu@ip-172-31-61-64:~/ansible-pythonapp-deployment$ ansible-playbook -i aws_ec2.yml --private-key=devops_demo.pem site.yml -u ec2-user
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.25.3) or chardet (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.25.3) or chardet (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)
[DEPRECATION WARNING]: 'include' for playbook includes. You should use 'import_playbook' instead. This feature will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.25.3) or chardet (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)

PLAY [all] *********************************************************************************************************************************************************************************************************

TASK [Update yum Cache] ********************************************************************************************************************************************************************************************
ok: [AnsibleManagedLoadBalancer]
ok: [AnsibleWebserver1]
ok: [AnsibleWebserver2]
ok: [AnsibleManagedDB]

PLAY [db] **********************************************************************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************************************************
ok: [AnsibleManagedDB]

TASK [mysql : download remi-mysql repo] ****************************************************************************************************************************************************************************
ok: [AnsibleManagedDB]

TASK [mysql : install remi-mysql repo] *****************************************************************************************************************************************************************************
 [WARNING]: Consider using the yum, dnf or zypper module rather than running 'rpm'.  If you need to use command because yum, dnf or zypper is insufficient you can add 'warn: false' to this command task or set 'command_warnings=False' in ansible.cfg to get rid of this
message.

changed: [AnsibleManagedDB]

TASK [mysql : install MySQL] ***************************************************************************************************************************************************************************************
[DEPRECATION WARNING]: Invoking "yum" only once while using a loop via squash_actions is deprecated. Instead of using a loop to supply multiple items and specifying `name: "{{item}}"`, please use `name: ['mysql', 'mysql-server']` and remove the loop. This feature will
be removed in version 2.11. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [AnsibleManagedDB] => (item=[u'mysql', u'mysql-server'])

TASK [mysql : run MySQL] *******************************************************************************************************************************************************************************************
```

5. Terraform

Terraform (code for infrastructure): a tool for building, changing and versioning infrastructure safely and efficiently.

Steps:

terraform code:instance.tf

```
provider "aws" {
access_key = ""
secret_key = ""
region     = "us-west-2"
}
resource "aws_instance" "tfmachine" {
ami =""
instance_type = "t2.micro"
}
```

step:

initialize the terraform where the terraform file resides:

```
#terraform init
#terraform plan ---says what it is going to do
#terraform plan -out tfplan.tf --plan save as tfplan.tf
#terraform apply tfplan.tf
#terraform destory --destroy the machine
```

But in enterprise level we create different tf files as shown below:

instance.tf--->

```
resource "aws_instance" "tfmachine" {
    instance_type = "t2.micro"
    ami = "${lookup(var.AMIS, var.AWS_REGION)}"
            subnet_id = ""
            vpc_security_group_ids = [" "]
    tags = {
        Name = "TFManagedMachine"
        ENV = "DEV"
    }

    provisioner "local-exec" {
        command = "echo ${aws_instance.tfmachine.private_ip} >> inventory.txt"
        }
}
output "ip" {
    value = "${aws_instance.tfmachine.public_ip}"
}
```

==========

provider.tf--->

```
provider "aws" {
        //access_key = "${var.AWS_ACCESS_KEY}"
        //secret_key = "${var.AWS_SECRET_KEY}"
        region     = "${var.AWS_REGION}"

        access_key = ""
        secret_key = ""
        }
```

vars.tf---->

```
variable "AWS_REGION" {
    type = "string"
    default = "us-west-2"
}
variable "AMIS" {
    type = "map"
    default = {
        us-west-2 = "ami-0e63f50857fdc1f9f"
        us-west-1 = "ami-0c1b880a476bb7b40"
```

```
        us-west-1 = "ami-0e818b29614c243bf"
}
}
```

backend.tf--->

```
terraform{
    backend "s3" {
    bucket = "tfstatebucket1"
    key = "terraform/tfstate"
    region = "us-west-2"
}
}
```

after using backend.tf terraform doesn't create terraform.tfstate file and the required file is created in the s3 bucket and the instance is being started up and running

```
      + password_data               = (known after apply)
      + placement_group             = (known after apply)
      + primary_network_interface_id = (known after apply)
      + private_dns                 = (known after apply)
      + private_ip                  = (known after apply)
      + public_dns                  = (known after apply)
      + public_ip                   = (known after apply)
      + security_groups             = (known after apply)
      + source_dest_check           = true
      + subnet_id                   = (known after apply)
      + tenancy                     = (known after apply)
      + volume_tags                 = (known after apply)
      + vpc_security_group_ids      = (known after apply)

      + ebs_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + snapshot_id           = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + iops                  = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

This plan was saved to: tfplan.tf

To perform exactly these actions, run the following command to apply:
    terraform apply "tfplan.tf"

[ec2-user@ip-172-31-58-30 first]$ terraform apply tfplan.tf
aws_instance.tfmachine: Creating...
aws_instance.tfmachine: Still creating... [10s elapsed]
aws_instance.tfmachine: Creation complete after 15s [id=i-008a409938f662cc5]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the `terraform show` command.

State path: terraform.tfstate
[ec2-user@ip-172-31-58-30 first]$ Connection reset by 54.202.180.21 port 22

Sabita Silwal@DESKTOP-HOUISRT MINGW64 ~
$ |
```

DriverUpdate™
There is a new version of DriverUpdate™ available. Would you like to update now?
No, Thanks Install Update

```
  Running scriptlet: vim-enhanced-2:8.0.1763-10.el8.x86_64                  4/4
  Running scriptlet: vim-common-2:8.0.1763-10.el8.x86_64                    4/4
  Verifying        : vim-enhanced-2:8.0.1763-10.el8.x86_64                  1/4
  Verifying        : vim-common-2:8.0.1763-10.el8.x86_64                    2/4
  Verifying        : gpm-libs-1.20.7-15.el8.x86_64                          3/4
  Verifying        : vim-filesystem-2:8.0.1763-10.el8.noarch                4/4

Installed:
  vim-enhanced-2:8.0.1763-10.el8.x86_64 vim-common-2:8.0.1763-10.el8.x86_64
  gpm-libs-1.20.7-15.el8.x86_64         vim-filesystem-2:8.0.1763-10.el8.noarch

Complete!
[ec2-user@ip-172-31-58-30 terraform_test]$ vim instance.tf
[ec2-user@ip-172-31-58-30 terraform_test]$ ls
first  instance.tf
[ec2-user@ip-172-31-58-30 terraform_test]$ mv instance.tf /first
mv: cannot move 'instance.tf' to '/first': Permission denied
[ec2-user@ip-172-31-58-30 terraform_test]$ sudo instance.tf /first
sudo: instance.tf: command not found
[ec2-user@ip-172-31-58-30 terraform_test]$ ls
first  instance.tf
[ec2-user@ip-172-31-58-30 terraform_test]$ sudo mv instance.tf /first
[ec2-user@ip-172-31-58-30 terraform_test]$ ls
first
[ec2-user@ip-172-31-58-30 terraform_test]$ cd first/
[ec2-user@ip-172-31-58-30 first]$ terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.
[ec2-user@ip-172-31-58-30 first]$ ls
[ec2-user@ip-172-31-58-30 first]$ cd ..
[ec2-user@ip-172-31-58-30 terraform_test]$ ls
first
[ec2-user@ip-172-31-58-30 terraform_test]$ find instance.tf
find: 'instance.tf': No such file or directory
[ec2-user@ip-172-31-58-30 terraform_test]$ ls
first
[ec2-user@ip-172-31-58-30 terraform_test]$ cd first/
[ec2-user@ip-172-31-58-30 first]$ vim instance.tf
[ec2-user@ip-172-31-58-30 first]$ ls
instance.tf
[ec2-user@ip-172-31-58-30 first]$ ls
instance.tf
[ec2-user@ip-172-31-58-30 first]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (terraform-providers/aws) 2.14.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.14"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-58-30 first]$
```

DriverUpdate™
There is a new version of DriverUpdate™ available. Would you like to update now?
No, Thanks Install Update

CronJobs:- important for devops engineer to schedule the jobs automatically at specified intervals.

STEPS:

#crontab -u root -e --->user as a root -e-->edit permissions

OR, You can go to below command to check the cronjobs

vim /var/spool/cron/root

write a command you need to execute

syntax is:

* * * * * /path/to/command (command) arg1 arg2 ..

```
- - - - -
| | | | |
| | | | ----- Day of week (0- 7) (Sunday=0 or 7)
| | | ------- Month (1- 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0- 23)
------------- Minute (0- 59)
```

for eg.

* * * * * /bin/uptime >>uptime.txt

it creates a file uptime.txt and updates the file every minute

--list all the cron jobs

crontab -l

or,

crontab -u root -l

NOTE:you can also run the script giving the path to that file with some arguments
eg. jenkins' workspace filled up very quickly so deleting the logs older than 30 days can be removed
that can be done using cron jobs

You can delete your cron jobs using:
#crontab -u root -r

NOTE=> There are shortcuts like @hourly, @monthly etc which can be easier
==========

6. Docker

Link: https://hub.docker.com/u/festivo1

Differences between contanerization and virtualization, docker and virtualmachine?

--virtualmachine – locks the resources

--docker containers—free the resources if not used

Steps:

for latest docker on ubuntu instance--

```
#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 install docker-ce docker-ce-cli containerd.io
```

docker is installed in ubuntu. you can check either by:

1. sudo systemctl status docker

2. docker --version

3. sudo docker run hello-world

Let's download ubuntu images from the dockerhub.com using pull command

#sudo docker pull ubuntu

#sudo docker run -it ubuntu bash --> -it--> interactive and provides the new bash for the new ubuntu containerd

#docker images ---> shows the latest images downloaded

#docker ps --> only running container

#docker ps -a --> gives all the container running or stopped ones

#docker stop [container_id] --> to stop the running container

for eg. You can pull one images

#docker pull kmlchauhan/centos7-jdk8-mvn3-jenkins:2.00 --> giving version too

After pulling you can run the images using:

#docker run -itd -p 8080:8080 --name=myjenkins1 kmlchauhan/centos7-jdk8-mvn3-jenkins:2.00

Note: i-interactive, t-terminal and d-daemon

If you don't specify d you will be stuck in that container and you have to kill that

Note: first port is external ip

second port for localhost

To login into your dockerhub account--> ask you for your username and password

#docker login

To enter the containers:

#docker exec -it myjenkins1 bash

#docker stop myjenkins1

#docker rm myjenkins1 --> to remove container myjenkins1 from the os

#docker rmi <Image_id or name> --> removes the docker images

#docker inspect <image_name>

running docker file to create image

you will have configuration files nginx.conf, runner.sh, etc

docker build -t [dockerhub_username]/nginx-helloworld:1.00 .

pushing the images to dockerhub account

#docker login --> give your credentials

docker push festivo1/nginx-helloworld:1.00

You can run your application as:

docker run -itd -p 80:80 --name=myapp festivo1/nginx-helloworld:1.00

Note: if you change your index.html, you need to build it again

After changing your html file you can directly run your container using:

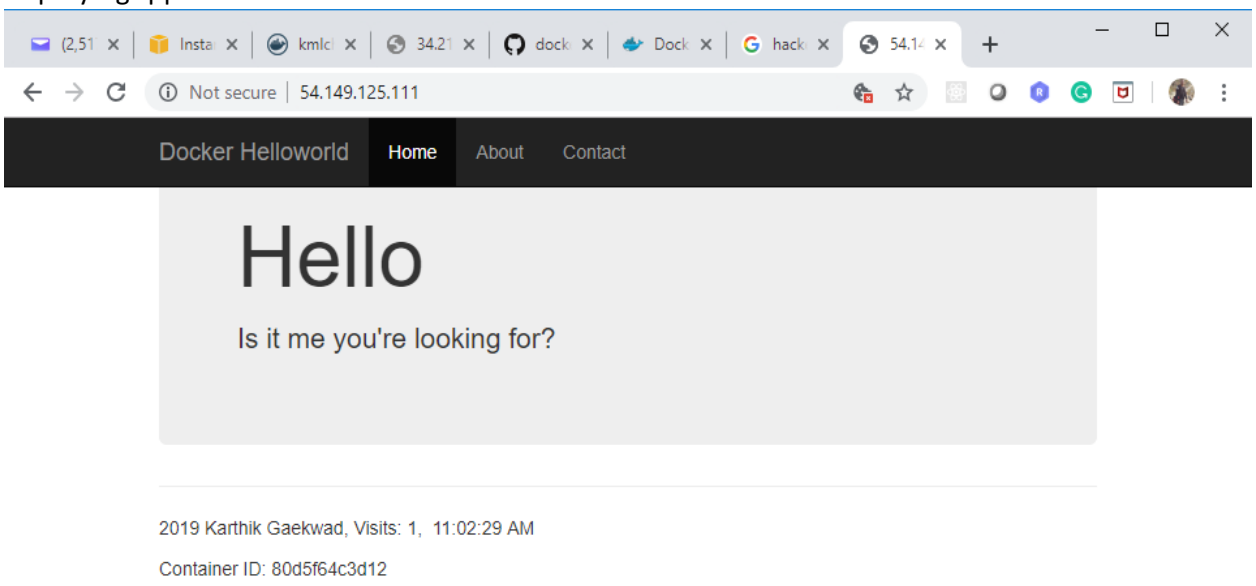
mapping the volume-->

docker run -itd -p 83:80 -v /root/docker/index.html:/www/data --name=myapp2 festivo1/nginx-helloworld:2

#docker kill <container_name> --> stops and removes at the same time

#docker commit --> creates a new image of the existing edited container

Deploying apps in docker:



Docker Compose:

install docker-compose

first install docker

#sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

#sudo chmod +x /usr/local/bin/docker-compose

#docker-compose --version

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the list of features.

Compose works in all environments: production, staging, development, testing, as well as CI workflows. You can learn more about each case in Common Use Cases.

Using Compose is basically a three-step process:

Define your app's environment with a Dockerfile so it can be reproduced anywhere.

Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.

Run docker-compose up and Compose starts and runs your entire app.

WORDPRESS example

-get wordpress compose file from docker documentation

```
-----------------

version: '3.3'


services:

  db:

    image: mysql:5.7
```

```
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER:  wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
   db_data: {}
```

---

save the above file in docker-compose.yaml

change the /etc/hosts file

127.0.0.1 db

127.0.0.1 wordpress

127.0.0.1 cache

run the command:

to check the docker volume:

#sudo docker volume ls

#docker-compose up -d

#docker volume ls

#docker network ls

#docker ps

#docker-compose kill

#docker volume rm <volume_name>

Docker Swarm:

----

install docker on each server and node

then on /etc/hosts define each pulic ip as master and slave on both the instance

then ssh:

ssh master (error will occur if you dont copy the id_rsa.pub key to the authorize file of another instance)

so do ssh-keygen in each instance and copy the id_rsa.pub key from each to authorized keys

do

ssh master from slave you can connect to master

do

ssh slave from master you can connect to slave

you can change your hostname going to /etc/hostname

then initialize docker swarm in manager

docker swarm init --advertise-addr <public_ip>

it becomes a manager

so save a command it gives to you as:

docker swarm join --token SWMTKN-1-5hcic3i95nq05u7g9v0w5z10zrv35eo5k51wqktaflu2e0nl9c-6os4u9ea7fbibnwx3fn65le30 52.12.77.82:2377

run the above command in your worker machine

docker swarm join --token SWMTKN-1-5r82qlkce19acj8qsv1p63cbfne9b0a3g1fy4xc4klma5y61sl-0qg2duc1ltxrn82n9oqze166h 54.213.240.170:2377

NOTE:

TCP port 2377 for cluster management communications

TCP and UDP port 7946 for communication among nodes

UDP port 4789 for overlay network traffic

docker node ls --check in master node

let's run some service:

docker service create --name my-web --publish 8080:80 nginx:1.13-alpine

docker service ls -- to check

Note: the services may run in any of the node(either master or slave)

7. KUBERNETES

==========----------

open source orchestration system for docker containers

minikube--

---

virtual box or vmware must be enabled

VT-x/AMD-v must be enabled

-go to bios setting -->advanced -->virtualization enbale--> save the changes

install chocolatey using cmd.exe using administrative access

put the command in cmd: @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

then check the installation using choco

then install minikube:

cmd:choco install minikube <provider>[optional] ==default is virtualbox

Now you can see minikube in your virutal box

from your command line:

kubectl get nodes

shows only master node

run the images:

kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080

--image=docker.io/nginx:latest --port=8080

kubectl get deployments

kubectl expose deployment hello-minikube --type=NodePort

kubectl get service

minikube service hello-minikube --url

minikube delete

```
operable program or batch file.

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>
C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>^Z^X
C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --por
t=8080
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --gene
rator=run-pod/v1 or kubectl create instead.
deployment.apps/hello-minikube created

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           76s

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           6m7s

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl expose deployment hello-minikube --type=NodePort
service/hello-minikube exposed

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get service
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
hello-minikube   NodePort    10.111.45.43   <none>        8080:30992/TCP   86s
kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP          55m

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>minikube service hello-minikube --url
http://192.168.99.100:30992

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   63m   v1.14.3

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl run nginx --image=docker.io/nginx:latest --port=80
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --gene
rator=run-pod/v1 or kubectl create instead.
deployment.apps/nginx created

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           55m
nginx            1/1     1            1           21s

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl expose deployment nginx --type=NodePort
service/nginx exposed

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>kubectl get service
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
hello-minikube   NodePort    10.111.45.43   <none>        8080:30992/TCP   11m
kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP          65m
nginx            NodePort    10.97.18.6     <none>        80:31384/TCP     35s

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>minikube service nginx --url
http://192.168.99.100:31384

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>minikube service hello-minikube --url
http://192.168.99.100:30992

C:\Users\Sabita Silwal\AppData\Local\Google\Cloud SDK>_
```

KOPs

kops(only works in linux or mac)

need to buy domain to work on it

8. Linux commands and bash scripting

Yum package management:

Yum --works with the repo configured on the system to install/remvoe/update package and thier dependencies

install =install a package from a repo
update =checks for updates and prompts to perform an update
remove= uninstalls a package
search=queries a repo for a package

for eg.

1.yum search httpd //if there is repo name httpd then it searches if not no match found
2. yum info httpd // it gives the detail about the package
3. yum install httpd //it installs the package to our system
4. yum list installed httpd // checks whether httpd is installed or not
5. yum deplist httpd //list the dependencies of httpd
6. yum remove httpd // to remove the httpd package not the dependencies
7. yum autoremove httpd // to remove httpd and only unnecessary dependencies
from teh system
8. which httpd //after removing no httpd in the system
9. yum list installed httpd // no matching package after removing
10. yum repolist // how many repo name and package
details of repolist are kept in files in fedora and centos in /etc/yum.repos.d folder
cd /etc/yum.repos.d //show some files
less CentOS-Base.repo //shows various urls for downloading the packages

11. yum clean all // clean all the packages
12. yum update // update the newer versions of the packages
13. download dig network commands from:
sudo yum install bind-utils
14. Networking net-tools: sudo yum install net-tools
15. rsync and scp commands--rsync only copy the changed amount of data from one host to another
but scp commands whole file again.

Networking commands:
-ifconfig -- to check ip address assigned to the system
-ip addr-- shows ip address and mac address
-traceroute-- print the route packets time to network host(list all the routers)
eg. traceroute google.com

-dig -- (domain information groper) tool for interrogating dns name server. It performs teh dns lookups
and displays the answers that are returned from the name servers.
eg. dig amazon.com
-telnet--telnet connect destination host:port via a telnet protocol if connection establishes means connectivity between
two hosts is working fine
eg. telnet google.com 443
-nslookup-- nslookup is a program to query internet domain name servers
eg. nslookup amazon.com
-netstat-- allows you a simple way to review each of your network connections and open sockets
netstat with head output is very helpful while performing web server troubleshooting.
eg. netstat
-scp and rsync-- allows you to secure copy files to and from another host in the network
eg. scp $filename user@targethost:/$path → normally path is tmp folder
-w -- prints a summary of the current activity on teh system, including what each user is doing and their processes.
eg. w
-nmap--powerful commands checks the opened port on the server
eg. nmap localhost
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
5432/tcp open postgresql
-enable/disable network interface
ifup eth0
ifdown eth0

LINUX FILE SYSTEM

/boot = file required for booting
/dev = Device file
/etc = config files
/home = home directory
/lib64 - libraries 64 bit
/lib - libraries for 32 bit
/media - modern directory
where removable media mounts
/mnt - where temp file systems are mounted
/opt - optional software
/var - variables

/run - runtime variable
/proc - processor (CPU)

To see the first lines

head -n <number of rows> <filename>

To see the last lines

tail -n <number of lines> <filename>

tail -f filename

filename =< 500MB

filename.1

tail -F filename → continuously show rows

ps aux → list out processes

Some linux scripting is pushed in my github account like installing Jenkins using bash, ruby, and python.

githubLink: https://github.com/festivo1/devops_tools