0. Infinite TOC

Friday, May 19, 2023 4:35 PM

Day 1: • Introduction to Ansible • What is Ansible and its key features • Ansible architecture • Installing Ansible • Installing Ansible on different operating systems • Configuring Ansible • Ansible Inventories • Creating inventory files • Host and group variables • Dynamic inventories • Ad-hoc Commands • Running ad-hoc commands using Ansible • Basic command syntax • Ansible modules	**Day 2:** • Ansible Playbooks • What are playbooks and why we need them • YAML syntax and structure of playbooks • Writing a basic playbook • Ansible Roles • Creating and using roles • Best practices for organizing roles • Ansible Vault • Encrypting sensitive data with Ansible Vault • Creating and managing vault files
Day 3: - <u>Additional topics to be added in the content:</u> • Jinja2 Templating • Create own custom modules • Dynamic inventories	

1. Introduction to Ansible

Friday, May 19, 2023 4:41 PM

What is Ansible and its key features?

- Ansible is an **open-source automation** tool that is used for configuration management, application deployment, and task automation.
- It is designed to be **simple, flexible, and powerful**, with a focus on ease of use and readability.

Some of the key features of Ansible include:

- **Agentless Architecture**: Ansible is an agentless tool, which means that you don't need to install any agents or software on the managed nodes. Instead, Ansible uses SSH to connect to the nodes and run commands.

- **YAML Syntax**: Ansible uses a simple and easy-to-understand YAML syntax for defining playbooks, which makes it easy for non-technical users to create and understand automation scripts.

- **Playbooks**: Ansible uses playbooks to define the automation tasks that need to be performed. Playbooks are written in YAML and can be used to perform a wide variety of tasks, including configuring servers, deploying applications, and managing infrastructure.

- **Idempotency**: Ansible is idempotent, which means that running the same playbook multiple times will always result in the same outcome, regardless of the state of the system. This makes it easy to perform automated tasks without worrying about unintended consequences.

- **Task Execution**: Ansible uses a task-based model for executing automation tasks. Each task is defined in a playbook, and Ansible executes the tasks in the order they are defined.

- **Inventory Management**: Ansible uses an inventory file to define the managed nodes and their properties. The inventory file can be in a variety of formats, including INI and YAML, and can be managed

dynamically using plugins.

- **Ad-hoc Commands**: Ansible allows you to run ad-hoc commands to perform quick tasks on managed nodes without having to create a playbook. Ad-hoc commands can be run from the command line or from within a playbook.

2. Ansible architecture

Friday, May 19, 2023 4:38 PM

Ansible has a client-server architecture that is designed to be simple, flexible, and powerful.
The Ansible architecture consists of several key components, including:

- **Control Node**: The control node is where *Ansible is installed* and where the Ansible playbooks are developed and executed. It can be a physical or virtual machine, and it runs the Ansible command-line tool.

- **Managed Nodes**: The managed nodes are the *servers, network devices, or other infrastructure that Ansible manages.* Ansible uses SSH to connect to these nodes and run commands.

- **Inventory**: The inventory is a *file that contains a list of the managed nodes* and their properties. It can be in a variety of formats, including INI and YAML, and can be managed dynamically using plugins.

- **Modules**: Modules are the units of work that Ansible uses to perform tasks on the managed nodes. Ansible has a large library of modules for performing tasks such as *installing packages, managing users, and copying files.*

- **Playbooks**: Playbooks are the files that *contain the automation tasks* that Ansible performs. Playbooks are written in YAML and can be used to perform a wide variety of tasks, including configuring servers, deploying applications, and managing infrastructure.

- **Task Execution**: Ansible uses a task-based model for executing automation tasks. Each task is defined in a playbook, and Ansible *executes the tasks in the order they are defined*.

- **API**: Ansible also has an API that allows developers to integrate Ansible into their own applications or workflows.

3. Installing Ansible

Friday, May 19, 2023 4:41 PM

 # Install EPEL repo (Centos/RedHat)
yum install epel-release

 # Install ansible package (Centos/RedHat)
yum install -y ansible

 # Upgrade & update (ubuntu)
apt upgrade -y && apt update -y
init 6

 # Install the software-properties-common package (ubuntu)
apt install software-properties-common

 # Install ansible personal package archive (ubuntu)
apt-add-repository ppa:ansible/ansible

 # Install ansible (ubuntu)
apt update && apt install ansible

4. Creating SSH key with ED22519 algorithm

Friday, May 19, 2023 4:41 PM

 # Create a new directory (ansible) and an inventory file (inventory) in it:
mkdir ansible
vim inventory
192.168.1.5
192.168.1.6
:wq!

 # Creating a secured SSH key using ED25519 algo & with a comment "ansible"
ssh-keygen -t ed25519 -C "ansible"

 # Copying the SSH key to Node 1
ssh-copy-id -i /home/jeetu/.ssh/id_ed25519.pub 192.168.1.5

 # Copying the SSH key to Node 2
ssh-copy-id -i /home/jeetu/.ssh/id_ed25519.pub 192.168.1.6

 # Verify the access (smooth)
ssh 192.168.1.5
ssh 192.168.1.6

 # Checking installed ansible version
ansible --version

Getting started: https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html#get-started-playbook

Playbook intro: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_int

ro.html#playbooks-intro

Using Ubuntu:

root@ubuntu-vm-0:~# **adduser ansibleuser**
Adding user `ansibleuser' ...
Adding new group `ansibleuser' (1001) ...
Adding new user `ansibleuser' (1001) with group `ansibleuser' ...
Creating home directory `/home/ansibleuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ansibleuser
Enter the new value, or press ENTER for the default
Full Name []: **Ansible user**
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] **Y**
root@ubuntu-vm-0:~# **su - ansibleuser**
ansibleuser@ubuntu-vm-0:~$

############################## ERROR ##################################
"Server refused our key" Only from MobaXterm
The functionality of these old keys can be restored by adding
PubkeyAcceptedKeyTypes +ssh-rsa
to /etc/ssh/sshd_config and restarting sshd.

5. Pinging servers

Friday, May 19, 2023 4:42 PM

 # Pinging all hosts within inventory (created above)
ansible -i inventory all -m ping #shortcut version

ansible all --key-file ~/.ssh/ansible -i inventory -m ping #actual version
ls--key-file = file path for ssh private key

Pinging specific host (192.168.1.5)
ansible -i inventory all -m ping --limit 192.168.1.5

6. Creating and copying a file to all remote servers

Friday, May 19, 2023 4:42 PM

 # Create a new dummy file with some (rough) content in it at /tmp

```
cat > /tmp/randomFile.txt
```

 # Copy this /tmp/randomFile.txt file to all the hosts within inventory file at /tmp:

```
ansible -i inventory all -m copy -a "src=/tmp/randomFile.txt dest=/tmp/randomFile.txt"
```

7. Installing finger package on ubuntu

Friday, May 19, 2023 4:42 PM

 # Installing package (finger) on ubuntu server group (finger-pkg)

ansible -i inventory finger-pkg -m apt -a "name=finger state=present" --become --ask-become-pass

 # Installing package (finger) on CENTOS server group (finger-pkg)

ansible -i inventory finger-pkg -m yum -a "name=finger state=present" --become --ask-become-pass

 # Uninstalling package (finger) on CENTOS server group (finger-pkg)

ansible -i inventory finger-pkg -m yum -a "name=finger state=absent" --become --ask-become-pass

here:

-i	inventory file path
-m	module name
-a "name=finger state=present	passes the package name ("finger") and the desired state ("present" meaning installed) to the apt module.
--become	become sudo user
--ask-become-pass	ask sudo user password
finger-pkg	Is the group name that needs to be added in the inventory file.

8. Creating local config file for ansible command shortening

Friday, May 19, 2023 4:42 PM

Creating local config file for ansible command shortening:

```
vim ansible.cfg                 #no change in name
    [defaults]
    inventory = inventory
    private_key_file = ~/.ssh/id_ed25519
:wq!
```

Testing these short commands with the help of local config file:

```
# Tesing the short cfg file
    ansible all -m ping
    ansible all --list-hosts
    ansible all -m gather_facts                         # gathering information
    ansible all -m gather_facts --limit <IP-address>   # targeting specific server.
    ansible all -m file -a "dest=/path/to/the/location mode = 777 owner = user1 group = user1 state = directory"
```

The following command checks if yum package is installed or not, but does not update it.

```
    # ansible all -m yum -a "name = package-name state = present"
```

The following command check the package is not installed.

```
    # ansible all -m yum -a "name = package-name state = absent"
```

The following command checks the latest version of package is installed.

```
    # ansible all -m yum -a "name = package-name state = latest"
```

Running other ad-hoc commands:

```
# Ad-hoc command to reboot all servers
    ansible all -m shell -a 'sudo shutdown -r now'
```

9. Creating inventory

Friday, May 19, 2023 4:43 PM

- Ansible automates tasks on managed nodes or “hosts” in your infrastructure, using a list or group of lists known as inventory.
- The simplest inventory is a single file with a list of hosts and groups.
- The default location for this file is **/etc/ansible/hosts**.
  - You can specify a different inventory file at the command line using the -i <path> option or in configuration using inventory.
- Here are three options/formats beyond the /etc/ansible/hosts file:
  - You can create a directory with multiple inventory files. (link)
  - You can pull inventory dynamically. (link)
  - You can use multiple sources for inventory, including both dynamic inventory and static files. (link)

Basic inventory file	Default groups	Hosts in multiple groups	Grouping groups: parent/child group relationships

`all:`
`  hosts:`
`  mail.example.com:`
`  children:`
`  webservers:`
`  hosts:`
`  foo.example.com:`
`  bar.example.com:`
`  dbservers:`
`  hosts:`
`  one.example.com:`
`  two.example.com:`
`  three.example.com:`
	• Even if you do not define any groups in your inventory file, Ansible creates two default groups: all and ungrouped.
	• The all group contains every host. The ungrouped group contains all hosts that don’t have another group aside from all.
		`all:`
		`  hosts:`
		`  mail.example.com:`
		`  children:`
		`  webservers:`
		`  hosts:`
		`  foo.example.com:`
		`  bar.example.com:`
		`  dbservers:`
		`  hosts:`
		`  one.example.com:`
		`  two.example.com:`
		`  three.example.com:`
		`  east:`
		`  hosts:`
		`  foo.example.com:`
		`  one.example.com:`
		`  two.example.com:`
		`  west:`
		`  hosts:`
		`  bar.example.com:`
		`  three.example.com:`
		`  prod:`
		`  hosts:`
		`  foo.example.com:`
		`  one.example.com:`
		`  two.example.com:`
		`  test:`
		`  hosts:`
		`  bar.example.com:`
		`  three.example.com:`
			`all:`
			`  hosts:`
			`  mail.example.com:`
			`  children:`
			`  webservers:`
			`  hosts:`
			`  foo.example.com:`
			`  bar.example.com:`
			`  dbservers:`
			`  hosts:`
			`  one.example.com:`
			`  two.example.com:`
			`  three.example.com:`
			`  east:`
			`  hosts:`
			`  foo.example.com:`
			`  one.example.com:`
			`  two.example.com:`
			`  west:`
			`  hosts:`
			`  bar.example.com:`
			`  three.example.com:`
			`  prod:`
			`  children:`
			`  east:`
			`  test:`
			`  children:`
			`  west:`

10 Playbook Scripts Demo

Thursday, March 14, 2024 2:01 PM

cat inventory (with a group)

```
[jeetu@cli01 ansible1]$ cat inventory
[server]
192.168.88.137
```

Creating default config
cat ansible.cfg

```
[jeetu@cli01 ansible1]$ cat ansible.cfg
[defaults]
inventory = inventory
private_key_file = /home/jeetu/.ssh/id_ed25519
[jeetu@cli01 ansible1]$
```

Playbook 1 : list all disk info
cat disk_inventory.yml

```
[jeetu@cli01 ansible1]$ cat disk_information.yml
---
- name: Gather disk information
  hosts: server
  gather_facts: yes
  tasks:
    - name: Display disk information
      debug:
        msg: "{{ ansible_devices['sda'] }}"
[jeetu@cli01 ansible1]$
[jeetu@cli01 ansible1]$ _
```

Code:

```
---
- name: Gather disk information
  hosts: server
  gather_facts: yes

  tasks:
    - name: Display disk information
      debug:
        msg: "{{ ansible_devices['sda'] }}"
```

Playbook 2 : list all installed package

List all with names & versions

```
---
- name: List installed packages on CentOS
  hosts: server
  tasks:
    - name: Get installed packages
      yum:
        list: installed
      register: installed_packages

    - name: Display installed packages
      debug:
        msg: "{{ installed_packages }}"

----------------------------------
# ansible-playbook -i inventory <file-name>.yml
```

List only names

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


---
- name: List installed packages on CentOS 7 and older
  hosts: server
  gather_facts: yes

  tasks:
    - name: Get installed package list
      ansible.builtin.shell: yum list installed | awk '{print $1}'
      register: installed_packages

    - name: Display installed package list
      debug:
        msg: "{{ installed_packages.stdout_lines }}"
----------------------------------
# ansible-playbook -i inventory <file-name>.yml
```

Playbook 3 : creating a user on remote machine(s)

1. Create encrypted password for the YAML.
   Open gitbash to encrypt:
   `# openssl passwd -1 -stdin <<< pass@word1`

2. Cat create_user.yml

```
---
- name: Manage users
  hosts: server
  become: yes

  tasks:
    - name: Create user
      ansible.builtin.user:
        name: username
        password: <$1$cqwvuVtX$6kTCPtlzMXExirchK.YSm/>
        state: present
```

Playbook 4 : creating a directory & file with custom permissions

```
[jeetu@cli01 ansible1]$ cat newdir.yml
---
- name: Create directory
  hosts: server
  become: yes

  tasks:
    - name: Create directory
      ansible.builtin.file:
        path: /jeetu-rocks
        state: directory

    - name: Create file
      ansible.builtin.file:
        path: /jeetu-rocks/myfile.txt
        state: touch
        mode: "0660"
        owner: jeetu
        group: jeetu
```

To run: ansible-playbook newdir.yml --become --ask-become-pass

```
[jeetu@cli01 ansible1]$ cat newdir.yml
---
- name: Create directory
  hosts: server
  become: yes

  tasks:
    - name: Create directory
      ansible.builtin.file:
        path: /jeetu-rocks
        state: directory

    - name: Create file
      ansible.builtin.file:
        path: /jeetu-rocks/myfile.txt
        state: touch
        mode: "0660"
        owner: jeetu
        group: jeetu
[jeetu@cli01 ansible1]$ ansible-playbook newdir.yml --become --ask-become-pass
```

Playbook 5 : Installing packages in bulk.

```
---
- name: Install packages
  hosts: server
  become: yes

  tasks:
    - name: Install required packages
      ansible.builtin.package:
        name: "{{ item }}"
        state: present
      loop:
```

```
    - finger
    - squid
    - httpd
```

To run: ansible-playbook install_bulk_pkgs.yml --become --ask-become-pass

Playbook 6 : Copying file from local to remote

```
---
- name: Copy files
  hosts: server
  become: yes

  tasks:
    - name: Copy files from local to remote
      ansible.builtin.copy:
        src: /README.txt
        dest: /README.txt
```

Playbook 7 : Copying file from remote to local

```
---
- name: Copy file from remote to local
  hosts: all
  gather_facts: no

  tasks:
    - name: Fetch file from remote machine
      ansible.builtin.fetch:
        src: /home/jeetu/remote.txt
        dest: /home/jeetu/remote.txt
        flat: yes
```

Script (GitHub): https://github.com/jitendrastomar5593/ansible-scripts

11. Creating inventory file in a different way

Friday, May 19, 2023 4:43 PM

METHOD - 1:
---

jeetu@ctrl:~/ansible$ vim new_inventory

```
    virtualmachines:
      hosts:
        node1:
          ansible_host: 192.168.1.5
        node2:
          ansible_host: 192.168.1.6
    :wq!
```

Output:

```
jeetu@ctrl:~/ansible$ ansible virtualmachines -m ping -i new_inventory
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

METHOD - 2:
---

jeetu@ctrl:~/ansible$ cat playbook.yml

```
- name: My custom script
  hosts: ubuntu
  tasks:
   - name: Ping my hosts
     ansible.builtin.ping:
   - name: Print message
     ansible.builtin.debug:
       msg: Hello world
```

Output:
---

ansible-playbook -i new_inventory playbook.yml

```
PLAY [My custom script] ********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [node2]
ok: [ctrl]
ok: [node1]

TASK [Ping my hosts] ***********************************************************************************************************
ok: [node2]
ok: [node1]
ok: [ctrl]

TASK [Print message] ************************************************************************************************************
ok: [ctrl] => {
    "msg": "Hello world"
}
ok: [node1] => {
    "msg": "Hello world"
}
ok: [node2] => {
    "msg": "Hello world"
}

PLAY RECAP *********************************************************************************************************************
ctrl                       : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node1                      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

12. Variables in ansible inventory

Friday, May 19, 2023 4:43 PM

In Ansible, an inventory is a file containing a list of target hosts, grouped into different categories or variables. Variables can be used to define host-specific configuration settings, group-specific configurations, or global configurations that apply to all hosts.

- Inventory file with variables in it.

```
jeetu@ctrl:~/ansible$ cat inventory_variables
[ubuntu]
192.168.1.5
192.168.1.6

[all:vars]
ansible_user=jeetu
ansible_ssh_private_key_file=/home/jeetu/.ssh/id_ed25519
```

- Output:

```
jeetu@ctrl:~/ansible$ ansible ubuntu -i inventory_variables -m ping
192.168.1.6 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.5 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

13. Ansible Modules

Friday, May 19, 2023 4:43 PM

Listing all modules in ansible:

- ansible-doc --list

```
add_host
amazon.aws.aws_az_facts
amazon.aws.aws_az_info
amazon.aws.aws_caller_facts
amazon.aws.aws_caller_info
amazon.aws.aws_s3
amazon.aws.cloudformation
amazon.aws.cloudformation_facts
amazon.aws.cloudformation_info
amazon.aws.ec2
amazon.aws.ec2_ami
amazon.aws.ec2_ami_facts
amazon.aws.ec2_ami_info
amazon.aws.ec2_eni
amazon.aws.ec2_eni_facts
amazon.aws.ec2_eni_info
amazon.aws.ec2_group
amazon.aws.ec2_group_facts
amazon.aws.ec2_group_info
amazon.aws.ec2_instance
amazon.aws.ec2_instance_facts
amazon.aws.ec2_instance_info
amazon.aws.ec2_key
amazon.aws.ec2_metadata_facts
amazon.aws.ec2_snapshot
amazon.aws.ec2_snapshot_facts
amazon.aws.ec2_snapshot_info
amazon.aws.ec2_spot_instance
amazon.aws.ec2_spot_instance_info
amazon.aws.ec2_tag
amazon.aws.ec2_tag_info
amazon.aws.ec2_vol
.
```

Listing details about specific module:

- ansible-doc <module_name>
- ansible-doc gather_facts
- ansible-doc ping

```
jeetu@ctrl:~/ansible$ ansible-doc  gather_facts
> ANSIBLE.BUILTIN.GATHER_FACTS    (/usr/lib/python3/dist-packages/ansible/modules/gather_facts.py

        This module takes care of executing the configured facts modules, the default is to use t
        This module is automatically called by playbooks to gather useful variables about remote
        It can also be executed directly by `/usr/bin/ansible' to check what variables are availa
        `facts' about the system, automatically.

ADDED IN: version 2.8 of ansible-core

  * note: This module has a corresponding action plugin.

OPTIONS (= is mandatory):

- parallel
        A toggle that controls if the fact modules are executed in parallel or serially and in or
        order of module facts at the expense of performance.
        By default it will be true if more than one fact module is used.
        [Default: (null)]
        type: bool


ATTRIBUTES:

        action:
          description: Indicates this has a corresponding action plugin so some parts of the
            options can be executed on the controller
          support: full
        async:
          description: Supports being used with the `async' keyword
          details: multiple modules can be executed in parallel or serially, but the action
            itself will not be async
```

14. Dynamic inventory with Azure

Friday, May 19, 2023 4:43 PM

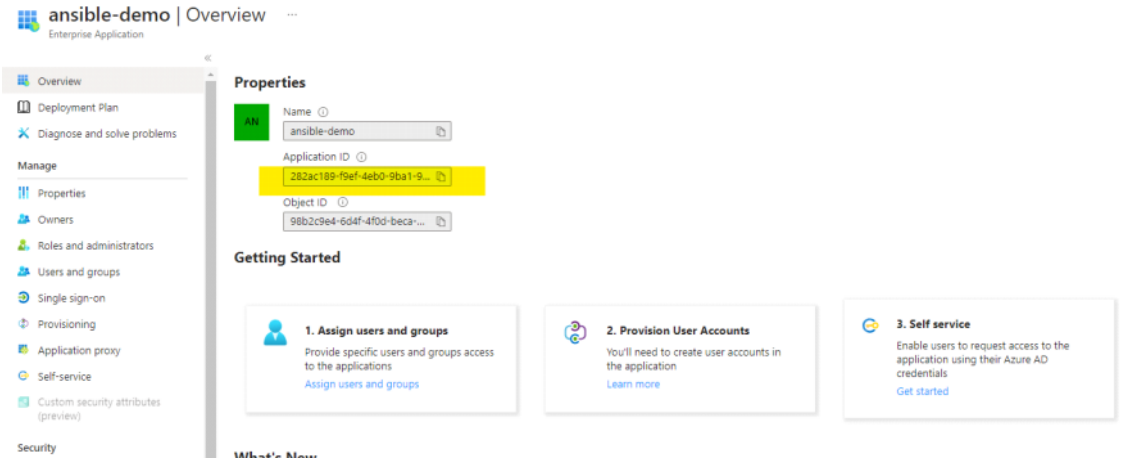
Link: [Tutorial: Configure dynamic inventories of your Azure resources using Ansible]

Link: [https://learn.microsoft.com/en-us/azure/developer/ansible/create-ansible-service-principal?tabs=azure-cli]

1. **Create an Azure service principal:**
   Command: az ad sp create-for-rbac --name ansible --role Contributor --scopes /subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b

```
PS /home/jitendra> az ad sp create-for-rbac --name ansible-demo --role Contributor --scopes /subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b
Creating 'Contributor' role assignment under scope '/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b'
The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into
{
  "appId": "282ac189-f9ef-4eb0-9ba1-994547e5204f",
  "displayName": "ansible-demo",
  "password": "G0b8Q~We4y1AlRac9w5A~6hONvnnq~UJB~qB5c-7",
  "tenant": "5659fac0-8e34-40af-86b2-dfcd9b0ddbf3"
}
PS /home/jitendra>
```

2. **Verifying:**



3. **Assign a role to the Azure service principal**:
   Command: az role assignment create --assignee 282ac189-f9ef-4eb0-9ba1-994547e5204f --role Contributor --scope /subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b

```
PS /home/jitendra> az role assignment create --assignee 282ac189-f9ef-4eb0-9ba1-994547e5204f --role Contributor --scope /subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b
{
  "condition": null,
  "conditionVersion": null,
  "createdBy": "658a6e7f-a4fa-40e3-ba6e-e73a3f9b5b7e",
  "createdOn": "2023-04-14T09:47:37.911272+00:00",
  "delegatedManagedIdentityResourceId": null,
  "description": null,
  "id": "/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b/providers/Microsoft.Authorization/roleAssignments/db81d7d2-597f-4130-9f33-e6acfb542334",
  "name": "db81d7d2-597f-4130-9f33-e6acfb542334",
  "principalId": "98b2c9e4-6d4f-4f0d-beca-9097a5b6d04d",
  "principalName": "282ac189-f9ef-4eb0-9ba1-994547e5204f",
  "principalType": "ServicePrincipal",
  "roleDefinitionId": "/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
  "roleDefinitionName": "Contributor",
  "scope": "/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b",
  "type": "Microsoft.Authorization/roleAssignments",
  "updatedBy": "658a6e7f-a4fa-40e3-ba6e-e73a3f9b5b7e",
  "updatedOn": "2023-04-14T09:47:37.911272+00:00"
}
PS /home/jitendra>
```

4. **Verifying the SP:**
   Command: az ad sp list --display-name ansible-demo --query '{clientId:[0].appId}'

```
PS /home/jitendra> az ad sp list --display-name ansible-demo --query '{clientId:[0].appId}'
{
  "clientId": "282ac189-f9ef-4eb0-9ba1-994547e5204f"
}
PS /home/jitendra>
```

5. **Install AZ-CLI on Ubuntu:**

sudo apt install azure-cli -y
az login
ansible-inventory -i myazure_rm.yml --graph

```
jeetu@ctrl:~/ansible$ ansible-inventory -i myazure_rm.yml --graph
@all:
  |--@ungrouped:
  |  |--ansible-controller_481c
  |  |--ansible-node1_fadb
  |  |--ansible-node2_f8cb
jeetu@ctrl:~/ansible$
```

6. **Fetching the VMs detailed info:**

Command: ansible-inventory -i myazure_rm.yml --list

```
jeetu@ctrl:~/ansible$ ansible-inventory -i myazure_rm.yml --list
{
    "_meta": {
        "hostvars": {
            "ansible-controller_481c": {
                "ansible_host": "40.88.37.30",
                "availability_zone": null,
                "computer_name": "ansible-controller",
                "data_disks": [],
                "default_inventory_hostname": "ansible-controller_481c",
                "id": "/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15
tualMachines/ansible-controller",
                "image": {
                    "offer": "0001-com-ubuntu-server-focal",
                    "publisher": "canonical",
                    "sku": "20_04-lts-gen2",
                    "version": "latest"
```

7. **Assign group membership with conditional_groups**

Open the myazure_rm.yml dynamic inventory and add the following conditional_group:

#vim myazure_rm.yml

```
plugin: azure_rm
include_vm_resource_groups:
 - ansible-inventory-test-rg
auth_source: auto
conditional_groups:
  linux: "'CentOS' in image.offer"
  windows: "'WindowsServer' in image.offer"
```

:wq!

Command: ansible-inventory -i myazure_rm.yml --graph

```
jeetu@ctrl:~/ansible$ ansible-inventory -i myazure_rm.yml --graph
@all:
  |--@ungrouped:
  |  |--ansible-controller_481c
  |  |--ansible-node1_fadb
  |  |--ansible-node2_f8cb
```

15. Dynamically fetching inventory from Azure RG:

Friday, May 19, 2023 4:44 PM

1. Login to Azure portal:
   # az login

2. Create a file (ansible_azure_rm.yml) anywhere:

```
jeetu@ctrl:~/ansible$ cat ansible_azure_rm.yml
plugin: azure_rm

include_vm_resource_groups:
- ansible-rg
auth_source: auto
```

   Output:

```
jeetu@ctrl:~/ansible$ ansible all -m ping -i ansible_azure_rm.yml
ansible-controller_481c | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible-node2_f8cb | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible-node1_fadb | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

   But, this is listing NIC card name, instead of VM name. To list actual VM names:

```
jeetu@ctrl:~/ansible$ cat ansible_azure_rm.yml
plugin: azure_rm

include_vm_resource_groups:
- ansible-rg
auth_source: auto
plain_host_names: yes
```

Output:

```
jeetu@ctrl:~/ansible$ ansible all -m ping -i ansible_azure_rm.yml
ansible-node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible-controller | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Reference link: https://www.shudnow.io/2019/12/12/ansible-dynamic-inventories-in-azure-part-1/

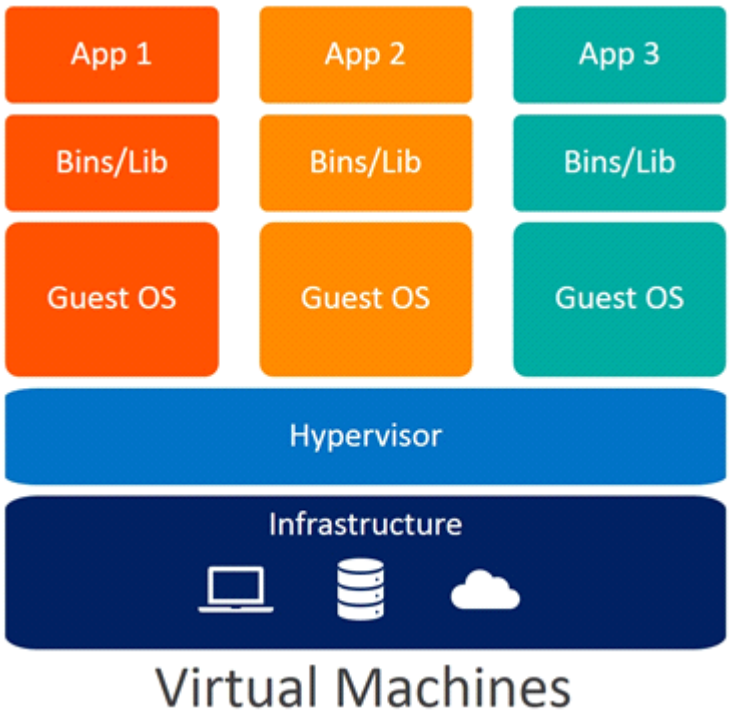
1. What is Virtualization?

Friday, May 19, 2023 4:44 PM

- Virtualization is the technique of importing a Guest operating system on top of a Host operating system.
- This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host.
- This eliminates the need for extra hardware resource.

What is a Virtual Machine?

- Virtual machines are heavy software packages that provide complete emulation of hardware devices like CPU, Disk and Networking devices.
- Virtual machines may also include a complementary software stack to run on the emulated hardware.
- These hardware and software packages combined produce a fully functional snapshot of a computational system.



Virtual Machines

2. What is a container?

Friday, May 19, 2023 4:45 PM

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

- A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

- Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure.

- Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Containerization

- Containerization is the packaging together of software code with all it's necessary components like libraries, frameworks, and other dependencies so that they are isolated in their own "container.“

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

3. What is Docker?

Friday, May 19, 2023 4:45 PM

- Docker is a software platform that allows you to build, test, and deploy applications quickly.
- Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.
- Build and run an image as a container.
- Share images using Docker Hub.

4. What is a Docker container?

Friday, May 19, 2023 4:46 PM

- A Docker container is a lightweight, standalone, and executable software package that includes everything needed to run an application, such as code, libraries, system tools, and settings.

- Containers are created using Docker, an open-source platform for building, shipping, and running applications in containers.

- Containers are isolated from the underlying operating system and other applications running on the same host, which helps to prevent conflicts between different applications and ensures that they have access to the resources they need without interfering with other applications.

- Docker containers are based on images, which are templates that define the contents of a container.

- Docker images can be created manually or automatically using Dockerfiles, which are text files that specify the steps needed to build an image. Once an image is created, it can be used to create one or more containers, each of which runs a separate instance of the application.

Docker containers that run on Docker Engine:

- **Standard**: Docker created the industry standard for containers, so they could be portable anywhere
- **Lightweight**: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs
- **Secure**: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry

Docker creates simple tooling and a universal packaging approach that bundles up all application dependencies inside a container which is then run-on Docker Engine.

Docker Engine enables containerized applications to run anywhere consistently on any infrastructure, solving "dependency issues" for developers and operations teams, and eliminating the "it works on my laptop!" problem.

5. Virtual Machine vs Docker container

Friday, May 19, 2023 4:46 PM

Virtual machines (VMs) and Docker containers are both technologies used for running applications in an isolated environment, but they have some significant differences.

A virtual machine:

- is an emulation of a complete physical machine, including a full operating system, hardware resources, and an application.
- Each VM runs its own copy of the operating system and is isolated from the host machine and other VMs.
- This isolation makes VMs very secure, but also resource-intensive, as each VM requires its own copy of the operating system and resources, such as CPU, memory, and disk space.

A Docker, on the other hand:

- is a containerization technology
- that provides a way to run applications in isolated environments without the overhead of a full operating system.
- Docker containers share the same kernel as the host machine, which makes them much more lightweight and efficient than VMs. Docker containers are also more portable than VMs, as they can be easily moved from one environment to another.

Here are some key differences between VMs and Docker containers:

Resource utilization:

- VMs require more resources than Docker containers, as they run a complete operating system in each instance.
- Docker containers, on the other hand, share the same kernel as the host machine and only require resources for the application and its dependencies.

Isolation:

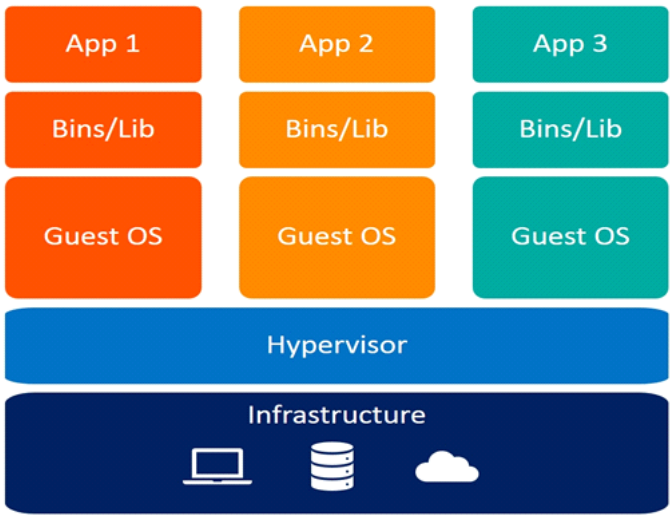
- VMs provide complete isolation from the host machine and other VMs, which makes them more secure, but also less efficient.
- Docker containers share the same kernel as the host machine and are more lightweight and efficient, but may not provide the same level of isolation as VMs.

Portability:

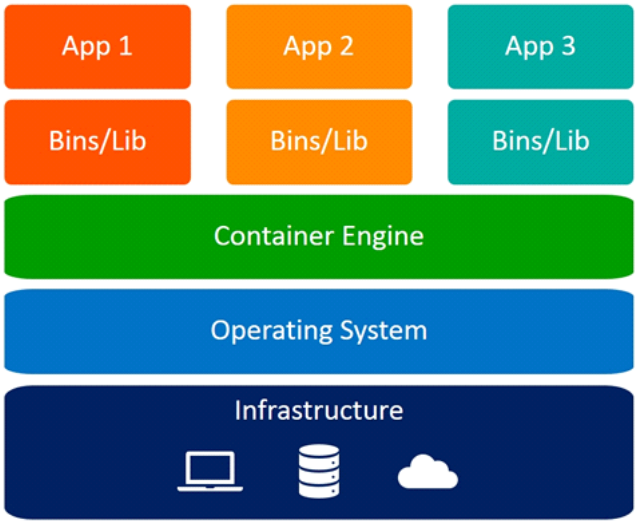
- Docker containers are more portable than VMs, as they can be easily moved from one environment to another.
- VMs, on the other hand, are less portable, as they may require different hardware or configurations in different environments.

Key	**Virtual Machine**	**Docker**
Resource utilization	VMs require more resources than Docker containers, as they run a complete operating system in each instance.	Docker containers, on the other hand, share the same kernel as the host machine and only require resources for the application and its dependencies.
Isolation	VMs provide complete isolation from the host machine and other	Docker containers share the same kernel as the host machine and are more

	VMs, which makes them more secure, but also less efficient.	lightweight and efficient, but may not provide the same level of isolation as VMs.
Portability	Docker containers are more portable than VMs, as they can be easily moved from one environment to another.	VMs, on the other hand, are less portable, as they may require different hardware or configurations in different environments.



Virtual Machines



Containers

6. When to use Containers?

Friday, May 19, 2023 4:46 PM

Containers are a good choice for the majority of application workloads. Consider containers in particular if the following is a priority:

Start time	• Docker containers typically start in a few seconds or less, whereas virtual machines can take minutes.
Efficiency	• Because Docker containers share many of their resources with the host system, they require fewer things to be installed in order to run. • A container typically takes up less space and consumes less RAM and CPU time. Due to this, you can fit more applications on a single server using containers. • containers may help to save money on cloud computing costs.
Licensing	• Most of the core technologies required to deploy Docker containers & Kubernetes, are free and open source, which is cost effective.
Code reuse	• Each running container is based on a container image, which contains the binaries and libraries that the container requires to run a given application. Container images are easy to build using Dockerfiles. • They can be shared and reused using container registries, which are basically repositories that host container images. • You can set up an internal registry to share and reuse containers within your company. • Thousands of prebuilt images can be downloaded from public registries (e.g. Docker Hub or Quay.io) for free and used as the basis for building your own containerized applications.

7. When to stick with virtual machines?

Friday, May 19, 2023 4:47 PM

Security	• Virtual machines are more isolated from each other and from the host system than are Docker containers. • Virtual machines are arguably more secure overall than containers.
Linux and Windows portability	• Docker is not as portable. Although in some ways Docker reduces dependence on your operating system. • Docker containers for Linux only work on Linux hosts, and the same holds true for Windows.
Rollback features	• Virtual machine platforms make it easy to “snapshot” virtual machines at a given point in time, and to “roll back” a machine when desired. • Docker doesn’t offer the same type of functionality. • You can roll back container images, but because containers store their data outside of the image in most cases, rolling back an image won’t help you recover data that was lost by a running application.

8. Why Switch to Docker?

Friday, May 19, 2023 4:47 PM

It works on my machine!!!

- This is common statement, given by the developer to another.
- This happens when you transfer or share the application you built to another person & he/she tries running the same application on their machines.
- The reason why this is happening are like:
  - One or more files missing.
  - Software version mismatch – MAJORILY OCCURING.
  - Different configuration settings.
- Solution is DOCKER.

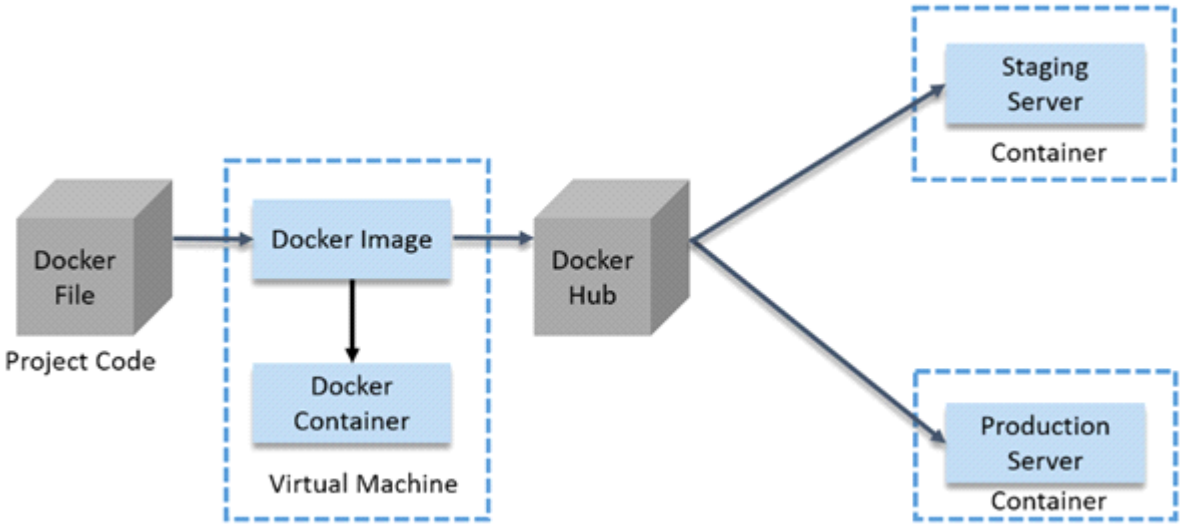
9. Terminologies of Docker

Friday, May 19, 2023 4:47 PM

1. **A Docker Image** - is created by the sequence of commands written in a file called as Dockerfile. When this Image is executed by “docker run” command it will by itself start whatever application or service it must start on its execution.
2. **Dockerfile** - When this Dockerfile is executed using a docker command it results into a Docker Image with a name.
3. **Docker Containers** - are a lightweight solution to Virtual Machines
4. **Docker Hub**
    - is like GitHub for Docker Images.
    - It is basically a cloud registry where you can find Docker Images uploaded by different communities or even by yourself.
5. **Docker Compose**
    - Docker Compose is basically used to run multiple Docker Containers as a single server.
6. **Docker Client**
    - The command line tool that allows the user to interact with the daemon.
7. **Docker Daemon**
    - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system which clients talk to.

10. How a Docker Container Works?

Friday, May 19, 2023 4:55 PM



1. A developer will first write the project code in a Docker file and then build an image from that file.
2. This image will contain the entire project code.
3. Now, you can run this Docker Image to create as many containers as you want.
4. This Docker Image can be uploaded on Docker hub (It is basically a cloud repository for your Docker Images, you can keep it public or private).
5. This Docker Image on the Docker hub, can be pulled by other teams such as QA or Prod.

Detailed work flow of a docker container?

1. **Dockerfile**: The process starts with creating a Dockerfile, which is a text file that contains instructions for building a Docker image. The Dockerfile specifies the base image, any additional dependencies, environment variables, and commands to run when the image is built.

2. **Docker Build:** Using the Docker CLI, you initiate the build process by running the docker build command and specifying the path to the directory containing the Dockerfile. Docker then reads the Dockerfile and builds an image according to the specified instructions.

3. **Image Layers**: During the build process, Docker uses a layered file system to optimize resource usage and enable efficient image sharing. Each instruction in the Dockerfile creates a new layer in the image. Layers are cached and can be reused in subsequent builds, speeding up the process.

4. **Docker Registry**: Once the image is built, it can be optionally pushed to a Docker registry. A registry is a centralized repository for storing and sharing Docker images. Common registries include Docker Hub, which is a public registry, and private registries that can be hosted locally or on cloud platforms.

5. **Docker Run**: To create and start a container from an image, you use the docker run command. This command specifies the image to use and any additional runtime configuration, such as port mappings, volume mounts, environment variables, and networking settings.

6. **Container Execution**: When the container is started, Docker creates an isolated environment based on the image's specifications. It sets up a separate file system, network namespace, and process space

for the container. The container runs a specific process or command defined in the image, and it can have its own set of resources allocated, such as CPU and memory.

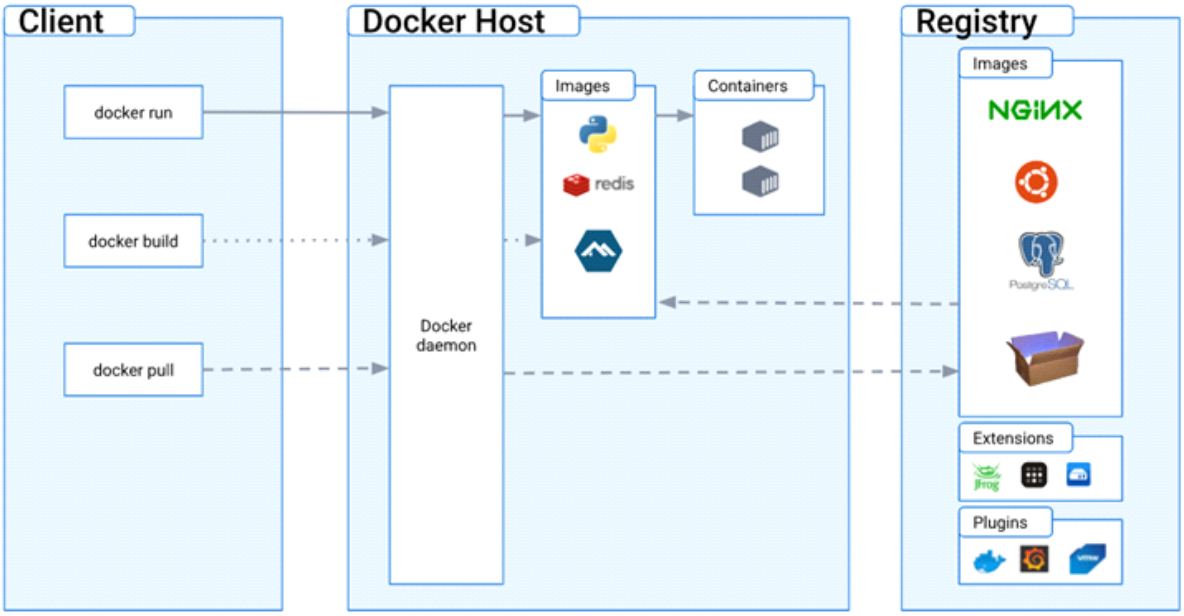
7. **Container Interaction**: Once the container is running, you can interact with it in various ways. You can access the container's console or shell, view its logs, copy files into or out of the container, and execute commands within the container using the Docker CLI.

8. **Container Lifecycle**: Containers can be paused, stopped, restarted, or removed using Docker commands. Pausing a container suspends its processes, while stopping it terminates them. Restarting a container restarts its processes. Removing a container deletes it and frees up the resources it was using.

9. **Container Orchestration**: In more complex scenarios, where multiple containers need to work together as part of a distributed application, container orchestration platforms like Docker Swarm or Kubernetes are used. These platforms provide advanced features such as service discovery, load balancing, scaling, and automated deployment of containers.

11. Docker architecture & components

Friday, May 19, 2023 5:14 PM



Docker architecture is composed of several components that work together to provide a platform for building, shipping, and running applications in containers. Here are the main components of the Docker architecture:

- Docker Engine
  - The Docker Engine is the core component of the Docker platform.
  - It's responsible for managing the lifecycle of containers, including starting, stopping, and restarting them.
  - The Docker Engine also provides an API and a command-line interface (CLI) for interacting with Docker.
- Docker images
  - Docker images are templates that define the contents of a container, including the application code, libraries, and system tools.
  - Docker images are created using a Dockerfile, which is a text file that specifies the steps needed to build the image.
- Docker containers
  - Docker containers are instances of Docker images that are running in an isolated environment.
  - Each container runs a separate instance of the application and has its own file system, network interface, and resource allocation.
- Docker registry
  - A Docker registry is a central repository where Docker images can be stored and shared between users.
  - Docker Hub is the default registry for Docker, but there are other options available, such as private registries that can be hosted on-premises or in the cloud.
- Docker CLI
  - The Docker CLI is a command-line interface that allows users to interact with the Docker Engine and perform common tasks, such as building images, running containers, and managing networks.
- Docker API

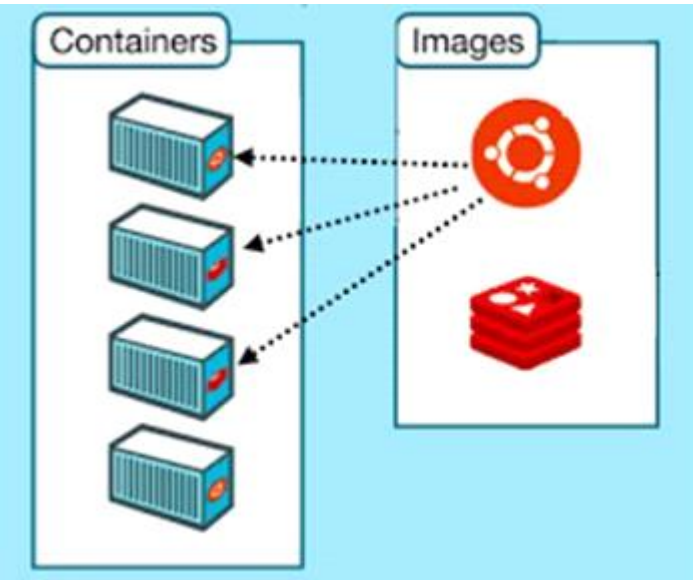
- The Docker API provides a programmatic interface for interacting with Docker. Applications can use the Docker API to automate the creation, deployment, and management of Docker containers.

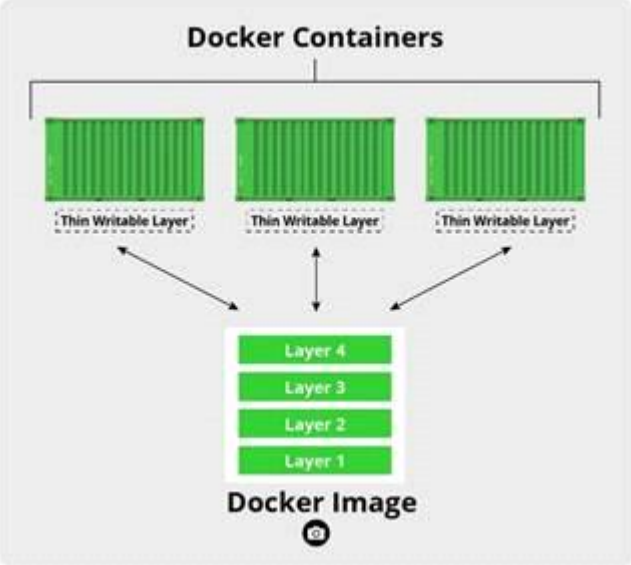
Explanation of each component in the diagram:

- The Docker daemon
  - The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
  - A daemon can also communicate with other daemons to manage Docker services.
- The Docker client
  - The Docker client (docker) is the primary way that many Docker users interact with Docker.
  - The client sends these commands to dockerd, which carries them out.
  - The docker command uses the Docker API.
  - The Docker client can communicate with more than one daemon.
- Docker Desktop (when your system/machine is old)
  - Docker Desktop is an easy-to-install application for your Mac, Windows or Linux environment that enables you to build and share containerized applications and microservices.
  - Docker Desktop includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.
- Docker registries
  - A Docker registry stores Docker images.
  - Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - You can even run your own private registry.
  - When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.
- Docker objects
  - When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

12. Images & Containers:

Friday, May 19, 2023 5:22 PM





Containers: contains application, environment.

Images: the templates/blueprint for container, an image contains code & app. Container runs this code. This image can create multiple times.

13. Docker Networking

Friday, May 19, 2023 5:23 PM

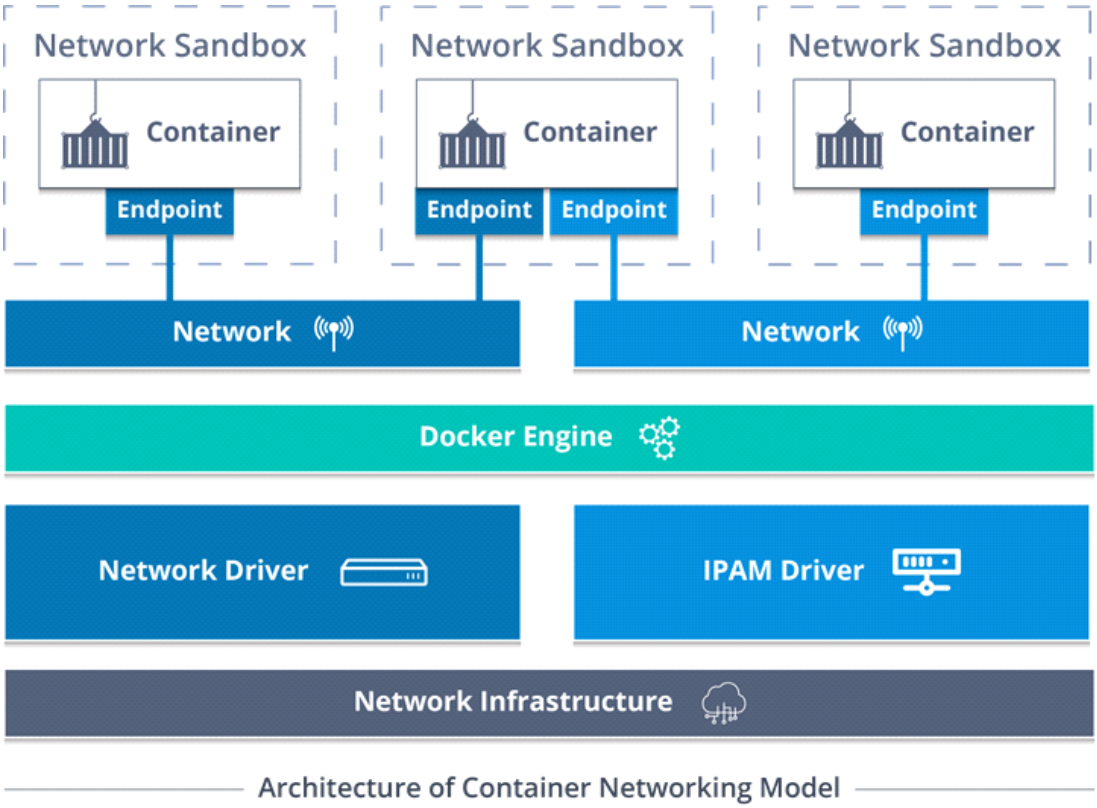
Docker Networking refers to the networking capabilities and features provided by Docker, an open-source containerization platform. Docker Networking allows containers to communicate with each other and with the outside world, enabling seamless connectivity and network integration for applications running in Docker containers.

Here are some key concepts and features related to Docker Networking:

- ✓ **Containers**:
  - Docker allows you to create and run containers, which are lightweight, isolated environments that package an application and its dependencies.
  - Each container can have its own network stack, including network interfaces, IP addresses, and routing tables.
- ✓ **Docker Network**:
  - A Docker network is a virtual network that provides communication between containers.
  - Docker offers various network drivers, such as bridge, overlay, host, and macvlan, each with its own characteristics and use cases.
  - By default, Docker creates a bridge network called "bridge" that allows containers to communicate with each other.
- ✓ **Bridge Network**:
  - The bridge network driver connects containers to a bridge on the host.
  - Containers attached to the same bridge can communicate with each other using IP addresses.
  - This is the default network driver for Docker containers and is suitable for most use cases.
- ✓ **Overlay Network**:
  - The overlay network driver allows containers to communicate across multiple Docker hosts.
  - It helps in the creation of distributed applications spanning multiple machines or even different data centers.
  - It uses a network overlay technology, such as VXLAN, to encapsulate and transmit network traffic between hosts.
- ✓ **Container-to-Container Communication**:
  - Containers within the same Docker network can communicate with each other using their IP addresses or container names.
  - Docker assigns each container a unique IP address within the network, allowing them to discover and communicate with other containers by their IP or hostname.
- ✓ **Exposing Ports**:
  - Docker allows you to expose ports on containers to the host system or external networks.
  - This enables access to containerized applications or services from the outside world. You can specify port mappings when running a container to bind a container port to a host port.
- ✓ **DNS Resolution**:
  - Docker provides an embedded DNS server that allows containers to resolve each other's IP addresses using container names.
  - This enables seamless service discovery and communication between containers without the need to know the IP addresses explicitly.

✓ **Docker Compose**:

- Docker Compose is a tool that simplifies the orchestration of multi-container applications.
- It provides a way to define and manage multiple containers as a single service using a YAML file.
- Docker Compose allows you to specify networking configurations, including network creation, container-to-container communication, and port mappings.



Architecture of Container Networking Model

✓ **Network sandbox**

- It is isolated sandbox that holds the network configurations of containers
- Sandbox is created when a user requests to generate an endpoint on the network.

✓ **Endpoints**

- It can have several endpoints in a network.
- EP established connectivity for the container services with other services.

✓ **Network**

- Provides networking components

✓ **Docker Engine**

- Is the base engine installed on host machine to build & run containers using docker.

✓ **Network driver**

- Its task is to manage the network with multiple drivers.

✓ **IPAM drivers**

- Manages the allocation and assignment of IP addresses to containers within a Docker network.

✓ **Network architecture**

- Provides connectivity between endpoints

IPAM driver:

An IPAM (IP Address Management) driver is a component that manages the allocation and assignment of IP addresses to containers within a Docker network.

The IPAM driver is responsible for the following tasks:

1. **IP Address Allocation**: The IPAM driver manages the pool of available IP addresses within a Docker network. When a container is created, the IPAM driver assigns an IP address to it from the available pool. This ensures that containers are assigned unique and non-conflicting IP addresses.
2. **Subnet Management**: The IPAM driver defines the subnet configuration for the Docker network, including the range of IP addresses that can be assigned to containers. It ensures that the IP addresses allocated to containers fall within the defined subnet.
3. **Gateway Configuration**: The IPAM driver sets up the default gateway for the Docker network. The gateway is the IP address used by containers to reach external networks. The IPAM driver assigns the appropriate gateway IP address to containers when they are connected to the network.
4. **Custom IP Address Assignment**: Some IPAM drivers allow for custom IP address assignment, where you can specify a particular IP address to be assigned to a container. This can be useful when you require specific IP address management for certain containers.
5. **Integration with External IPAM Systems**: Docker IPAM drivers can integrate with external IP Address Management systems, such as IPAM systems used in large-scale network deployments. This allows Docker to leverage existing IPAM infrastructure and management processes.

14. Installing Docker on Centos 7

Friday, May 19, 2023 5:30 PM

1. Install docker commands: (run as root user)

---

yum install -y yum-utils
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

- docker-ce
- docker-ce-cli
- containerd.io
- docker-buildx-plugin
- docker-compose-plugin

yum repolist
yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
which docker

2. verify if docker is running, else start it.

---

systemctl status docker
systemctl start docker
systemctl enable docker
systemctl status docker

3. verify docker image & container status

---

docker info

4. checking images

---

listing images
[root@docmaster ~]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
[root@docmaster ~]# docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE

5. downloading the image(s)

---

docker pull ubuntu //this downloads latest ubuntu version
docker pull ubuntu:18.04 //this downloads specific ubuntu version

14-b. Installing Docker on Ubuntu (22.04)

Thursday, March 7, 2024 12:11 PM

OS requirements
To install Docker Engine, you need the 64-bit version of one of these Ubuntu versions:

- Ubuntu Mantic 23.10
- Ubuntu Jammy 22.04 (LTS)
- Ubuntu Focal 20.04 (LTS)

Install using the apt repository:

1. Set up Docker's apt repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2. Install docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

3. Verify that the Docker Engine installation is successfully:

```
docker run hello-world
```

Read more: https://docs.docker.com/engine/install/ubuntu/

15. Docker basic commands

Friday, May 19, 2023 5:39 PM

Here are some basic Docker commands that you can use to work with Docker containers:

docker run: Creates and runs a new container based on an image.
Example: - Runs an interactive Ubuntu container and opens a bash shell.
docker run -it ubuntu:latest /bin/bash

docker ps: Lists running containers.
Example: - Lists all running containers.
docker ps

docker images: Lists available Docker images.
Example: - Lists all locally available Docker images.
docker images

docker pull: Downloads an image from a Docker registry.
Example: - Downloads the latest Nginx image from Docker Hub.
docker pull nginx:latest

docker build: Builds an image from a Dockerfile.
Example: - Builds an image named "myimage" using the Dockerfile in the current directory.
docker build -t myimage:latest .

docker stop: Stops a running container.
Example: - Stops the container with the specified ID.
docker stop container_id

docker start: Starts a stopped container.
Example: - Starts the container with the specified ID.
docker start container_id

docker restart: Restarts a running container.
Example: - Restarts the container with the specified ID.
docker restart container_id

docker rm: Removes a container.
Example: - Removes the container with the specified ID.
docker rm container_id

docker rmi: Removes an image.
Example: - Removes the image with the specified ID.
docker rmi image_id

docker exec: Executes a command inside a running container.
Example: - Executes an interactive bash shell inside the running container.
docker exec -it container_id /bin/bash

docker logs: Displays the logs of a container.
Example: - Shows the logs of the container with the specified ID.
docker logs container_id

16. Docker basic commands (with snapshots)

Friday, May 19, 2023 5:51 PM

I have already installed DOCKER CE on Ubuntu 20.04 on Azure VM.

To list docker images details (like: running, stopped, paused & downloaded images).
#docker info

```
root@dockerdemo:~# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.9.1-docker)
  compose: Docker Compose (Docker Inc., v2.12.2)
  scan: Docker Scan (Docker Inc., v0.21.0)

Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
```

To list current images within docker:
docker image ls

```
root@dockerdemo:~# docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
ubuntu       14.04     13b66b487594   19 months ago    196MB
```

To download latest docker images from DockerHub:
docker pull alpine

```
root@dockerdemo:~# docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
213ec9aee27d: Pull complete
Digest: sha256:bc41182d7ef5ffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Thinking-Why Alpine: *alpine is a minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size.

To verify if the image is downloaded:
docker image ls

```
root@dockerdemo:~# docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
alpine        latest    9c6f07244728   3 months ago     5.54MB
ubuntu        14.04     13b66b487594   19 months ago    196MB
root@dockerdemo:~#
```

To run the image as a container:
docker run -i -t alpine /bin/sh

```
root@dockerdemo:~# docker run -i -t alpine /bin/sh
/ # whoami
root
/ # hostname
efe08ac3ec31
/ # pwd
/
/ #
```

i = interaction
t = terminal access
alpine = image we downloaded
/bin/sh = default shell to write linux commands

To list docker images:
docker ps -a

```
root@dockerdemo:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS                       PORTS     NAMES
efe08ac3ec31   alpine         "/bin/sh"     2 minutes ago    Exited (0) 18 seconds ago              zealous_solomon
2b4b92f119f2   alpine         "/bin/bash"   3 minutes ago    Created                                objective_edison
cb72731032c9   alpine         "/bin/bash"   3 minutes ago    Created                                relaxed_elbakyan
2ba5f78f8f61   ubuntu:14.04   "/bin/bash"   17 hours ago     Exited (127) 17 hours ago              unruffled_williamson
root@dockerdemo:~#
```

To run another images:
docker run -it ubuntu /bin/bash

```
root@dockerdemo:~# docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
e96e057aae67: Pull complete
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Status: Downloaded newer image for ubuntu:latest
root@a5bd4d4b33ff:/#
```

Checking running containers:
docker ps -a

```
root@dockerdemo:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS                      PORTS     NAMES
a5bd4d4b33ff   ubuntu         "/bin/bash"   10 minutes ago   Exited (127) 14 seconds ago           youthful_meitner
efe08ac3ec31   alpine         "/bin/sh"     2 hours ago      Exited (0) 2 hours ago                zealous_solomon
2b4b92f119f2   alpine         "/bin/bash"   2 hours ago      Created                               objective_edison
cb72731032c9   alpine         "/bin/bash"   2 hours ago      Created                               relaxed_elbakyan
2ba5f78f8f61   ubuntu:14.04   "/bin/bash"   19 hours ago     Exited (127) 19 hours ago             unruffled_williamson
```

Starting & stopping docker container:
Starting →# docker start <container_id>
Stop →# docker stop <container_id>

```
root@dockerdemo:~# docker start a5bd4d4b33ff
a5bd4d4b33ff
root@dockerdemo:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS
a5bd4d4b33ff   ubuntu         "/bin/bash"   11 minutes ago   Up 4 seconds
efe08ac3ec31   alpine         "/bin/sh"     2 hours ago      Exited (0) 2 hours ago
2b4b92f119f2   alpine         "/bin/bash"   2 hours ago      Created
cb72731032c9   alpine         "/bin/bash"   2 hours ago      Created
2ba5f78f8f61   ubuntu:14.04   "/bin/bash"   19 hours ago     Exited (127) 19 hours ago
root@dockerdemo:~# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS              PORTS     NAM
a5bd4d4b33ff   ubuntu    "/bin/bash"   12 minutes ago   Up About a minute             you
root@dockerdemo:~# docker stop a5bd4d4b33ff
a5bd4d4b33ff
root@dockerdemo:~#
```

Get the docker stats while its running:
docker stats

```
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
a5bd4d4b33ff   youthful_meitner  0.00%     844KiB / 7.765GiB   0.01%     876B / 0B   0B / 0B     1
```

17. login to Docker hub

Friday, May 19, 2023 5:51 PM

To create a custom image, you need to login to docker (NOT MANDATORY):
docker login

```
root@dockerdemo:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to crea
e one.
Username: tomarj44
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@dockerdemo:~#
```

18. Creating 1st Dockerfile (basic)

Friday, May 19, 2023 5:52 PM

Creating Dockerfile from scratch:

#vim Dockerfile

```
root@dockerdemo:~# cat Dockerfile
FROM ubuntu:16.04
COPY . /var/www/html
root@dockerdemo:~#
```

Building docker image using Dockerfile:

docker build -t dockerdemo-website .

```
root@dockerdemo:~# docker build -t dockerdemo-website .
Sending build context to Docker daemon  25.6kB
Step 1/2 : FROM ubuntu:16.04
16.04: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:1f1a2d56de1d604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6
Status: Downloaded newer image for ubuntu:16.04
 ---> b6f507652425
Step 2/2 : COPY . /var/www/html
 ---> 03e0c3950141
Successfully built 03e0c3950141
Successfully tagged dockerdemo-website:latest
```

Listing created image:

docker image ls

```
root@dockerdemo:~# docker image ls
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
dockerdemo-website  latest    03e0c3950141   2 minutes ago    135MB
ubuntu              latest    a8780b506fa4   6 days ago       77.8MB
alpine              latest    9c6f07244728   3 months ago     5.54MB
ubuntu              16.04     b6f507652425   14 months ago    135MB
ubuntu              14.04     13b66b487594   19 months ago    196MB
root@dockerdemo:~#
```

Running the container, you just build:

docker run -it -d -p 80:80 dockerdemo-website

```
root@dockerdemo:~# docker run -it -d -p 80:80 dockerdemo-website
44447626cd93aa5462fab834207039f52108d18c2ca1bf95c8445325362c1405
```

Verifying the running container:

docker ps -a

```
root@dockerdemo:~# docker ps -a
CONTAINER ID   IMAGE              COMMAND       CREATED             STATUS                      PORTS
44447626cd93   dockerdemo-website "/bin/bash"   4 minutes ago       Up 4 minutes                0.0.0.0:80->80/tcp, :::80->80/tcp
a5bd4d4b33ff   ubuntu             "/bin/bash"   About an hour ago   Up 41 minutes
efe08ac3ec31   alpine             "/bin/sh"     3 hours ago         Exited (0) 3 hours ago
2b4b92f119f2   alpine             "/bin/bash"   3 hours ago         Created
cb72731032c9   alpine             "/bin/bash"   3 hours ago         Created
2ba5f78f8f61   ubuntu:14.04       "/bin/bash"   20 hours ago        Exited (127) 20 hours ago
root@dockerdemo:~#
```

19. Creating a new image using existing Ubuntu docker image

Friday, May 19, 2023 5:56 PM

Step1 – download/pull a new docker image from docker hub

docker pull ubuntu(already there for me)

```
root@dockerdemo:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1b
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Step2 – Run a new container (name - testUbuntu) with the downloaded ubuntu image & update it too

docker run --name testUbuntu -it ubuntu

```
root@dockerdemo:~# docker run --name testUbuntu  -it ubuntu
root@539abfb18af3:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [13.6 kB]
```

Step3 – verify if the container is present.

docker image ls

```
root@dockerdemo:~# docker image ls
REPOSITORY           TAG       IMAGE ID       CREATED          SIZE
dockerdemo-website   latest    03e0c3950141   6 hours ago      135MB
ubuntu               latest    a8780b506fa4   6 days ago       77.8MB
alpine               latest    9c6f07244728   3 months ago     5.54MB
hello-world          latest    feb5d9fea6a5   13 months ago    13.3kB
ubuntu               16.04     b6f507652425   14 months ago    135MB
ubuntu               14.04     13b66b487594   19 months ago    196MB
root@dockerdemo:~#
```

Step4 – Verify if GIT is installed or not

git –version

```
root@8431ab907f7f:/# git --version
bash: git: command not found
```

Step5 – Install GIT

apt-get install git

```
root@8431ab907f7f:/# apt-get install git
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
```

Step6 – Verify GIT version

git --version

```
root@8431ab907f7f:/# git --version
git version 2.34.1
```

Step7 – Exit the console

exit

Step8 – verify if the container is still running:

docker ps -a

```
root@dockerdemo:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED         STATUS                     PORTS   NA
MES
8431ab907f7f   ubuntu    "bash"    3 minutes ago   Exited (0) 7 seconds ago           my
Ubuntu
```

Step9 – check all current images:

docker images

```
root@dockerdemo:~# docker images
REPOSITORY           TAG       IMAGE ID       CREATED          SIZE
dockerdemo-website   latest    03e0c3950141   6 hours ago      135MB
ubuntu               latest    a8780b506fa4   6 days ago       77.8MB
alpine               latest    9c6f07244728   3 months ago     5.54MB
hello-world          latest    feb5d9fea6a5   13 months ago    13.3kB
ubuntu               16.04     b6f507652425   14 months ago    135MB
ubuntu               14.04     13b66b487594   19 months ago    196MB
```

Step10 - create your own new image using this container & verify

docker commit myUbuntu custom-ubuntu

docker images

```
root@dockerdemo:~# docker commit myUbuntu custom-ubuntu
sha256:292c55511acf46f6e275afa3bc4e444ea62316161a0fa1aef184a9d0edb239a8
root@dockerdemo:~# docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
custom-ubuntu       latest    292c55511acf   6 seconds ago    192MB
```

Step11 – Run another container using this same (custom-ubuntu) image

docker run -it --name test1 custom-ubuntu

```
root@dockerdemo:~# docker run -it --name  test1 custom-ubuntu
```

Step12 – verify GIT is present or not.

```
root@424c1b116098:/# git --version
git version 2.34.1
```

20. Creating a new image using Dockerfile

Friday, May 19, 2023 5:59 PM

Step1 – Create a "Dockerfile" using any editor.

vim Dockerfile

D in Dockerfile must be CAPITAL

```
FROM centos:7
MAINTAINER Jitendra jitendra@pioneerbusineessolutions.in
RUN yum makecache
RUN yum upgrade -y
RUN yum install -y httpd
RUN yum install -y git
RUN yum install -y vi
```

Step2 – build the docker image using the Dockerfile.

docker build -t test_centos:Docker .

```
root@dockerdemo:~# docker build -t test_centos:Dockerfile .
Sending build context to Docker daemon  30.72kB
Step 1/7 : FROM centos:7
7: Pulling from library/centos
2d473b07cdd5: Pull complete
Digest: sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa73129f60f4bf2bd22b407
Status: Downloaded newer image for centos:7
 ---> eeb6ee3f44bd
Step 2/7 : MAINTAINER Jitendra jitendra@pioneerbusineessolutions.in
 ---> Running in cc0920350806
Removing intermediate container cc0920350806
 ---> 71034dfbcc3f
```

Step3 – listing the custom-built image.

docker images

```
root@dockerdemo:~# docker images
REPOSITORY          TAG         IMAGE ID       CREATED          SIZE
test_centos         Dockerfile  92a2a12df8cc   20 minutes ago   910MB
custom-ubuntu       latest      292c55511acf   13 hours ago     192MB
dockerdemo-website  latest      03e0c3950141   19 hours ago     135MB
ubuntu              latest      a8780b506fa4   7 days ago       77.8MB
alpine              latest      9c6f07244728   3 months ago     5.54MB
hello-world         latest      feb5d9fea6a5   13 months ago    13.3kB
```

Step4 – running a new container using this custom container.

docker run -it --name cen_test_container test_centos:Dockerfile

```
root@dockerdemo:~# docker run -it --name cen_test_container test_centos:Dockerfile
[root@833293a52bb2 /]# rpm -q httpd git vim
httpd-2.4.6-97.el7.centos.5.x86_64
git-1.8.3.1-23.el7_8.x86_64
package vim is not installed
[root@833293a52bb2 /]# yum install -y vim
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: us.mirror.nsec.pt
 * extras: repos.lax.layerhost.com
 * updates: mirror.atl.genesisadaptive.com
Package 2:vim-enhanced-7.4.629-8.el7_9.x86_64 already installed and latest version
Nothing to do
[root@833293a52bb2 /]#
```

cen_test_container = new container name

test_centos:Dockerfile = passing existing docker image with ‘Dockerfile’ tag.

21. Pushing the docker image to docker hub

Friday, May 19, 2023 5:59 PM

Pushing the docker image to docker hub: - be ready with your custom username/image:tag

Step1 – login to docker hub using console

docker login

```
root@dockerdemo:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
e one.
Username: tomarj44
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Username: tomarj44

Password : XXX

Note: image must follow name as <docker hub username>/<image name>:<version>

Step2 – rename your image with appropriate convention:

docker tag test_centos:Dockerfile tomarj44/test_centos:Dockerfile

```
root@dockerdemo:~# docker tag test_centos:Dockerfile tomarj44/test_centos:Dockerfile
```

Verify:

```
root@dockerdemo:~# docker images
REPOSITORY              TAG         IMAGE ID       CREATED             SIZE
test_centos             Dockerfile  92a2a12df8cc   About an hour ago   910MB
tomarj44/test_centos    Dockerfile  92a2a12df8cc   About an hour ago   910MB
```

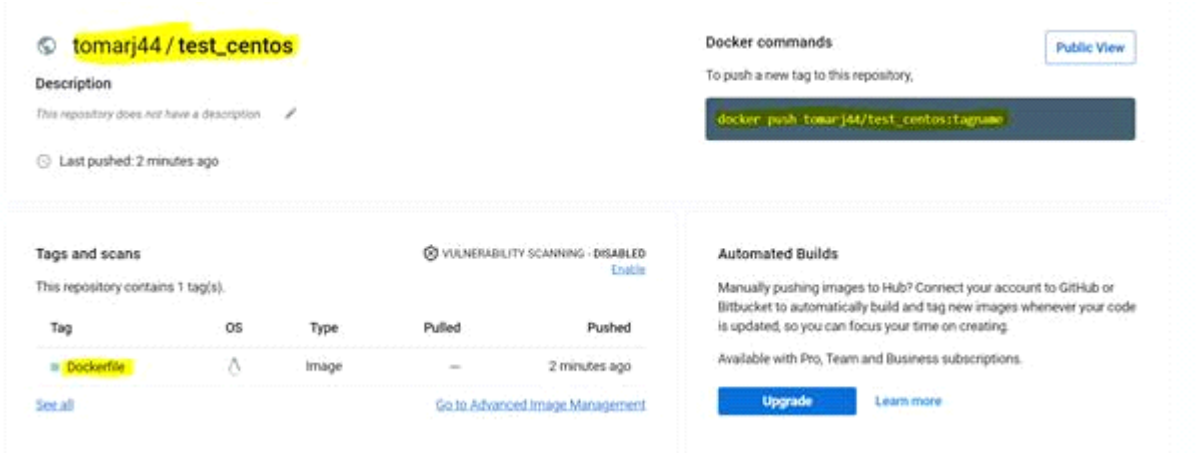
Step3 – Upload / Push your custom image on docker hub:

docker push tomarj44/test_centos:Dockerfile

```
root@dockerdemo:~# docker push tomarj44/test_centos:Dockerfile
The push refers to repository [docker.io/tomarj44/test_centos]
1cfe5bda793e: Pushed
fc577fe703f8: Pushed
a33817b811ed: Pushed
ae4fba6d1497: Pushed
b6c48e12cc2a: Pushed
174f56854903: Pushed
Dockerfile: digest: sha256:2cd51cb05d3591f747f63b956291ffae2fd82662db72882a6daf5582b66902c5 size: 1590
```

Step4 – Verify on docker hub:

Login to hub.docker.com



Step5 – download this image from another account

Login to VMWare W.S VM

docker pull tomarj44/test_centos:Dockerfile

```
[root@svr ~]# docker pull tomarj44/test_centos:Dockerfile
Dockerfile: Pulling from tomarj44/test_centos
2d473b07cdd5: Pull complete
553c605ca6a6: Pull complete
48caa1f7083b: Pull complete
1d378cb0610b: Pull complete
e85df4839226: Pull complete
5847aa9dbcb4: Pull complete
Digest: sha256:2cd51cb05d3591f747f63b956291ffae2fd82662db72882a6daf5582b66902c5
Status: Downloaded newer image for tomarj44/test_centos:Dockerfile
docker.io/tomarj44/test_centos:Dockerfile
[root@svr ~]#
[root@svr ~]# docker images
REPOSITORY              TAG          IMAGE ID       CREATED          SIZE
tomarj44/test_centos    Dockerfile   92a2a12df8cc   2 hours ago      910MB
hello-world             latest       feb5d9fea6a5   13 months ago    13.3kB
[root@svr ~]#
```

22. Testing App - 1st

Friday, May 19, 2023 6:02 PM

Dockerfile:

```
[root@docmaster docker-data]# cat Dockerfile
# Fetching the image
FROM centos:7

# Setting up the working dir
WORKDIR /app

# Running the command
RUN yum install -y httpd

# Copying content to required directory
COPY . /var/www/html/

# Allowing port 80
EXPOSE 80

#running the HTTPD command
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello, World!</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="hello">
    Hello, World!
  </div>
<marquee>This app is running using Docker</marquee >
</body>
</html>
```

styles.css

```
.hello {
  color: blue;
  font-size: 24px;
  text-align: center;
  margin-top: 50px;
}
```

Note: once image is built & then later you change the code, it won't reflect even you restart

new containers as images are read-only copy. You need to re-build the image, if the code is changed.

23. Docker Networking Commands

Friday, May 19, 2023 6:10 PM

list existing networks
docker network ls

create a new network
docker network create <mynetwork>

verify if its created
docker network ls

Create and attach a new container to the network
docker run --network=mynetwork --name=mycontainer -d ubuntu:18.04

rename the container, if new container needs to be created.
docker run --network=mynetwork --name=mycontainer2 -d ubuntu:18.04

inspecting container
docker ps or (-a) //fetch container ID
docker inspect <container-id> //check last lines
docker inspect <container-id> | grep mynetwork
docker inspect d57b75e0097c | grep IPAddress

running container with interaction & specific network
docker run -it --network=mynetwork --name=mycontainer2 ubuntu:18.04

to connect with 2 different,
Start an nginx container, give it the name 'mynginx' and run in the background
docker run --name mynginx --detach nginx

Get the IP address of the container
docker inspect mynginx | grep IPAddress

Run busybox (a utility container). It will join the bridge network
docker run -it busybox sh

go inside bustbox & run below command:
/ # wget -q -O - 172.17.0.2:80
##
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
.
.
.
.
.

```
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
########################################
```

24. Docker storage commands

Friday, May 19, 2023 6:11 PM

listing all docker volumes
docker volume ls

Create a new volume named 'my-vol'
docker volume create my-vol

listing all docker volumes, again
docker volume ls

inspecting volume
docker volume inspect my-vol

```
[
    {
        "CreatedAt": "2023-05-17T19:32:38+05:30",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/my-vol/_data",
        "Name": "my-vol",
        "Options": null,
        "Scope": "local"
    }
]
```

//The 'Mountpoint' is the location where the files or folders going to store actually.

creating a file with some data onto this mount point
cat > /var/lib/docker/volumes/my-vol/_data/demo.txt
this is demo by
Jeetu

attaching a single container to 2 containers & checking them.
docker run -it -v my-vol:/root --name my-container1 alpine
/ # cd /root/
~ # ls
demo.txt
~ # cat demo.txt
this is a demo volume file.
by
Jeetu

creating a new container with the same volume attached & checking the same file.
docker run -it -v my-vol:/root --name my-cont2 ubuntu
root@4a419cc5e7c0:/# cd /root/
root@4a419cc5e7c0:~# ls
demo.txt
root@4a419cc5e7c0:~# cat demo.txt
this is a demo volume file.
by
Jeetu

25. Docker Compose Installation step by step

Friday, May 19, 2023 6:11 PM

- Compose is a tool for defining and running multi-container Docker applications.
- Compose works in all environments: production, staging, development, testing, as well as CI workflows.
- It also has commands for managing the whole lifecycle of your application:
  - Start, stop, and rebuild services
  - View the status of running services
  - Stream the log output of running services
  - Run a one-off command on a service

Installing standalone docker-compose version (link):

Working shell script:
https://gist.github.com/deviantony/2b5078fe1675a5fedabf1de3d1f2652a

1. **To download and install Compose standalone,** run
   # curl -SL https://github.com/docker/compose/releases/download/v2.18.0/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
2. **Give executable permissions**:
   # chmod +x /usr/local/bin/docker-compose
3. **Verify:**
   # chmod +x /usr/local/bin/docker-compose
4. **Check/verify version**:
   # docker-compose version

To create a container using docker compose

1. Create a new folder (/root/docker-compose)
2. Change the directory to this folder
3. Create a file "**docker-compose.yaml**" (file name must be this only) & paste/write data given below:

```
services:
  mytest:
    image: hello-world
:wq!
```

```
services:
  mytest:
    image: hello-world
```

4. Then, run below command:
   # docker-compose up

5. Valid the presence:
   # docker ps -a

26. Docker-Compose-app-1

Friday, May 19, 2023 6:14 PM

Dockerfile content:	docker-compose.yml content	Index.php content

FROM php:7.4-apache
RUN docker-php-ext-install mysqli
COPY . /var/www/html/
	version: '3'
	services:
	  web:
	  build:
	  context: .
	  dockerfile: Dockerfile
	  ports:
	  - 80:80
	  depends_on:
	  - db
	  db:
	  image: mysql:5.7
	  ports:
	  - 3306:3306
	  environment:
	
	MYSQL_ROOT_PASSWORD: root
	  MYSQL_DATABASE: mydatabase
	  MYSQL_USER: myuser
	  MYSQL_PASSWORD: mypassword
		<?php
		$host = 'db';
		$user = 'myuser';
		$password = 'mypassword';
		$database = 'mydatabase';
		
		$conn = new mysqli($host, $user, $password, $database);
		
		if ($conn->connect_error) {
		  die('Connection failed: ' . $conn->connect_error);
		}
		
		$result = $conn->query('SELECT "Hello, MySQL!" AS message');
		$row = $result->fetch_assoc();
		$message = $row['message'];
		
		$conn->close();
		?>
		
		<!DOCTYPE html>
		<html>
		<head>
		  <title>PHP MySQL Example</title>
		</head>
		<body>
		 <?php echo $message; ?>
		</body>
		</html>

Run docker compose:

docker-compose up

In case, you re-edit the Dockerfile:

docker-compose build

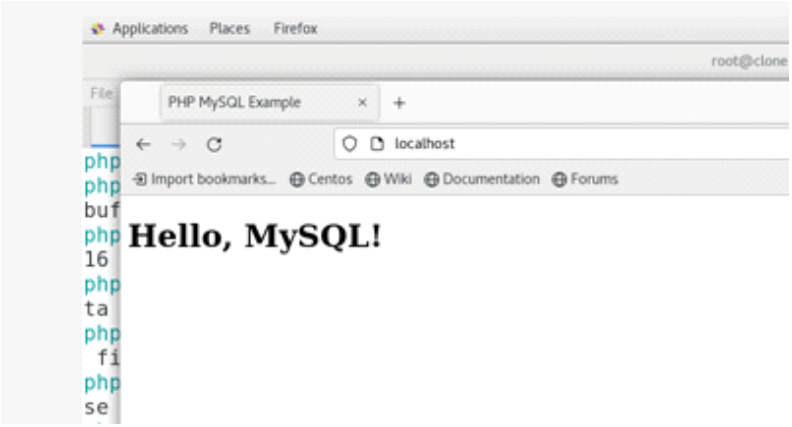
docker-compose up

To verify:

On web browser, http://localhost

To stop properly:

docker-compose down


Applications
Places
Firefox
root@clone
File
PHP MySQL Example
localhost
Import bookmarks...
Centos
Wiki
Documentation
Forums
Hello, MySQL!

27. Docker-compose-app-2

Wednesday, March 6, 2024 11:23 PM

1. **Create a directory for the project:**
   mkdir composetest
   cd composetest

2. **Create a file "app.py" in current location:**

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.
\n'.format(count)
```

3. **Create another file called "requirements.txt"**
   flask
   redis

4. **Create a Dockerfile**

```
# syntax=docker/dockerfile:1
FROM python:3.10-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

5. **Create a file called compose.yaml**

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

6. **Build and run your app with Compose**
   # docker compose up

7. **Enter http://localhost:8000/ in a browser to see the application running & refresh it to see the changes.**

8. **Check all the images:**
   # docker images

9. **To stop the application:**
   # docker compose down

Edit the Compose file to add a bind mount

10. **Edit the "compose.yaml"**

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_DEBUG: "true"
  redis:
    image: "redis:alpine"
```

11. **Re-build and run the app with Compose**
    # docker compose up

12. **Update the application.**
    Make changes in "app.py" & refresh your browser.

13. **Running the docker compose in "detached" mode:**
    # docker compose up -d
    # docker compose ps

14. **To stop docker compose**
    # docker compose stop

15. **Removing container with the data volume**
    # docker compose down --volumes

Link: https://docs.docker.com/compose/gettingstarted/
Sample code: https://github.com/docker/awesome-compose
How docker compose works: https://docs.docker.com/compose/compose-application-model/

Git Intro

Introduction to Git:
Git is a powerful and widely-used distributed version control system that enables developers to efficiently manage and track changes in their codebase. Originally created by Linus Torvalds in 2005, Git has become an essential tool for software development and collaboration.

Key Points:

1. **Version Control**: Git allows developers to keep track of changes made to their code over time, providing a history of modifications, who made them, and why.
2. **Distributed System**: Unlike centralized version control systems, Git is distributed, meaning every developer has a complete copy of the project's history on their local machine. This fosters collaboration and enables offline work.
3. **Branching**: Git makes it easy to create branches, enabling developers to work on isolated features or bug fixes without affecting the main codebase. This promotes parallel development and experimentation.
4. **Collaboration**: Git facilitates collaborative coding by providing tools for merging and resolving conflicts when multiple developers work on the same project simultaneously.
5. **Snapshots**, Not Changes: Git doesn't store just the changes between versions; it stores entire snapshots of the project at each commit. This ensures the integrity and reliability of your codebase.
6. **Open Source**: Git is open source, with a thriving community and a wealth of resources available for users. Platforms like GitHub, GitLab, and Bitbucket offer hosting services for Git repositories.
7. **Command Line and GUI**: Git can be used via a command-line interface (CLI) for advanced users or through user-friendly graphical user interfaces (GUIs) for those who prefer a visual approach.

Installation
https://git-scm.com/

Configuration
After installation, you should configure your name and email in Git. This information is used to identify your commits.

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Initializing a Repository
To start tracking changes in a project, navigate to the project's root directory and run:

```
git init
```

Cloning a Repository:
To clone an existing Git repository from a remote server (like GitHub or GitLab) to your local machine, use

```
git clone <repository_url>
```

Staging and Committing:
Git uses a two-step process to save changes:
Staging: You stage changes you want to include in the next commit using:

```
git add <file(s)>
```

Committing: Once changes are staged, you commit them with a message explaining the changes:

```
git commit -m "Your commit message"
```

Checking Status:

To see the status of your working directory (untracked files, modified files, etc.), use:

```
git status
```

Viewing Commit History:

To view the commit history of the repository, use:

```
git log
```

Branching:

Git allows you to work on different features or versions simultaneously using branches. To create a new branch, use:

```
git branch <branch_name>
```

To switch to a branch, use:

```
git checkout <branch_name>
```

To create and switch to a new branch in one command, use:

```
git checkout -b <branch_name>
```

Merging:

After working on a branch, you can merge it back into the main branch (e.g., "master") using:

```
git merge <branch_name>
```

Commonly used Git commands:

To track file using git :	# git init
To add all the files in staging area:	# git add --a (or) # git add .
To commit the code:	# git commit -m "Commit message"
To check status:	# git status
To list logs:	# git log
To unlink a folder from GIT:	# rm -rf .git

Git Command

1. git init:
- Description: Initializes a new Git repository in the current directory.
- Usage: Run this command in the root directory of your project to start version control.

2. git clone <repository_url>:
- Description: Creates a copy of a remote repository on your local machine.
- Usage: Replace <repository_url> with the URL of the remote repository you want to clone.

3. git add <file(s)>:
- Description: Stages changes for the next commit.
- Usage: You can specify specific files to stage or use git add . to stage all changes.

4. git commit -m "Your commit message":
- Description: Commits the staged changes with a descriptive message.
- Usage: Replace "Your commit message" with a brief explanation of the changes.

5. git status:
- Description: Displays the status of your working directory, including untracked files and modified files.
- Usage: Run this command to see the current state of your repository.

6. git log:
- Description: Shows a log of all commits in the current branch.
- Usage: Use this command to view the commit history, including commit messages, authors, and timestamps.

7. git branch <branch_name>:
- Description: Creates a new branch with the specified name.
- Usage: Replace <branch_name> with the desired name for your new branch.

8. git checkout <branch_name>:
- Description: Switches to the specified branch.
- Usage: Use this command to move between branches in your repository.

9. git checkout -b <branch_name>:
- Description: Creates a new branch and switches to it in one command.
- Usage: Replace <branch_name> with the desired name for your new branch.

10. git merge <branch_name>:
- Description: Combines the changes from one branch into another.
- Usage: Use this command while on the target branch to merge changes from <branch_name> into the current branch.

11. git remote add <remote_name> <remote_url>:
- Description: Adds a remote repository to your local Git configuration.
- Usage: Replace <remote_name> with a name for the remote (e.g., "origin") and <remote_url> with the URL of the remote repository.

12. git fetch <remote_name>:
- Description: Retrieves changes from a remote repository without merging them.
- Usage: Use this command to see what changes are available on the remote repository.

13. git pull <remote_name> <branch_name>:
- Description: Fetches changes from a remote repository and merges them into the current branch.
- Usage: Replace <remote_name> with the remote you want to pull from and <branch_name> with the branch to pull.

14. git push <remote_name> <branch_name>:
- Description: Pushes your local commits to a remote repository.
- Usage: Replace <remote_name> with the remote repository and <branch_name> with the branch to push.

Git Commands - step by step (working)

Checking GIT version:
git --version

Creating a directory
mkdir gitdemo2

Switching the dir:
cd gitdemo2/

Listing the current data:
ls

Creating a notepad file:
notepad demo.txt

Checking the current status:
git status

Initializing the GIT:
git init

Setting up the username:
git config --global user.name "Jeetu"

Setting up the email ID:
git config --global user.email "jitendrastomar5593@gmail.com"

Checking the default config: user.
git config --list

Initializing the GIT:
git init

Checking the status now:
git status

Adding the file(s) to the stagging
git add . //for multiple files
git add <file1> <file2>...

Checking the status again:
git status

Adding the remote GIT repo
git remote add origin https://github.com/jitendrastomar5593/gitdemo2.git

Setting up the main branch

git branch -M main

Committing the code
git commit -m "1st commit"

Pushing the data to git to main branch
git push -u origin main //OR
git push origin main

Checking the log:
git log

Creating a new file:
notepad 2ndfile.txt

Adding this file to stagging
git add 2ndfile.txt

Committing this code:
git commit -m "2nd commit"

Pushing the code GIT repo:
git push -u origin main

Checking the log again:
git log

Checking the remote:
git remote -v

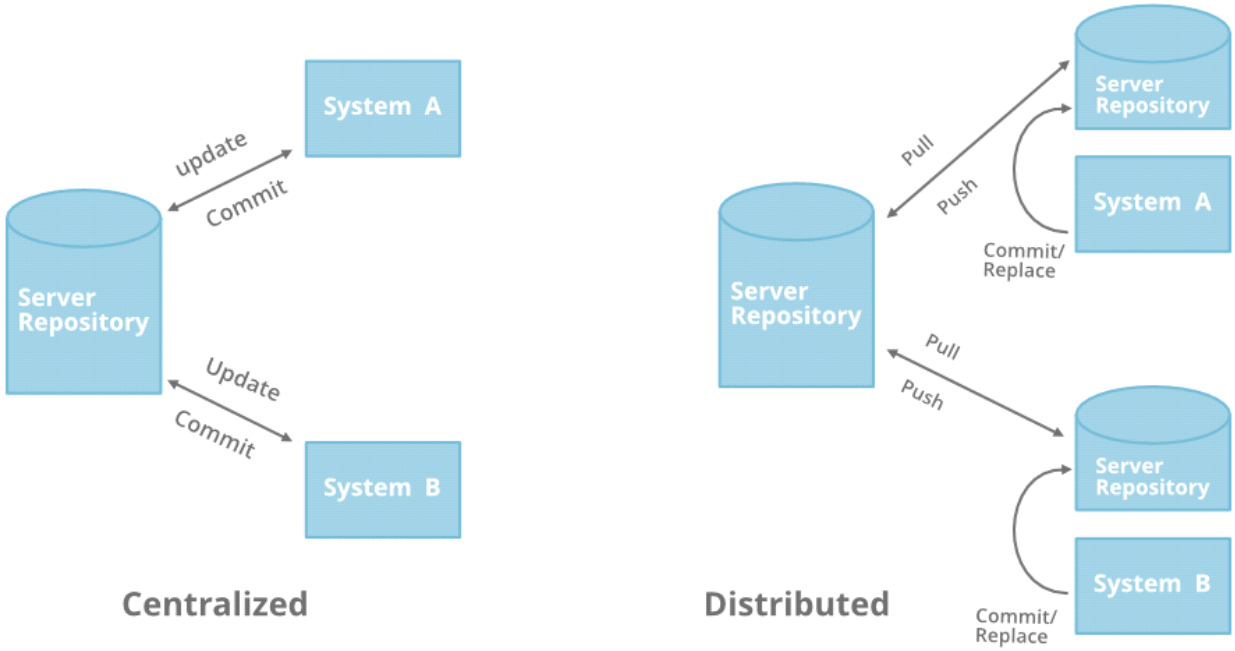
Changing the remote URL:
git remote set-url origin https://github.com/jitendrastomar5593/gitdemo2.git

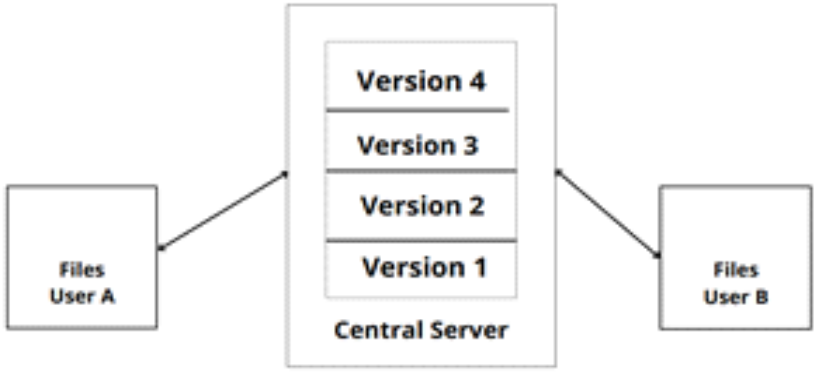
Checking the remote:
git remote -v

CVCS vs DVCS

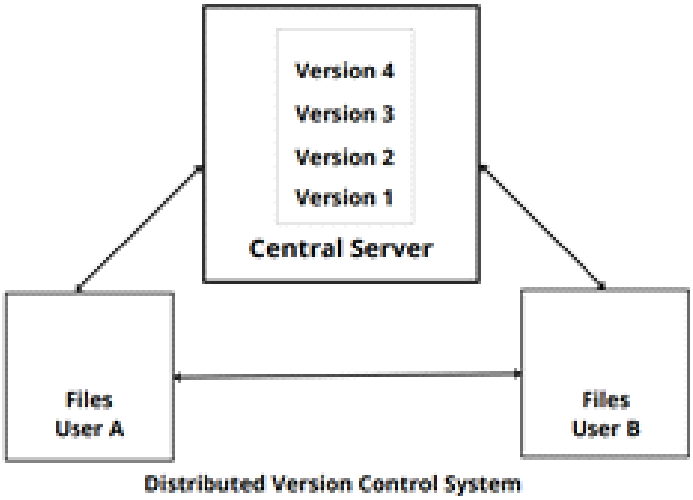
Centralized Version Control System	Distributed Version Control System
Repository Structure: • There is a central repository that stores the entire version history of the project. • Users typically have a working copy of the latest version of the files on their local machines. **Network Dependency:** • Requires a constant connection to the central server for most operations. • If the central server goes down or there are network issues, users may face difficulties in performing version control operations. **Branching and Merging:** • Branching and merging can be more challenging compared to DVCS. • Typically, branching involves creating a copy of the codebase on the central server, and merging often requires more careful coordination. **Speed:** • Speed of operations can be slower, especially when the central repository is large or when there are many users accessing it simultaneously. **History:** • The entire history of the project is stored centrally.	**Repository Structure:** • Each user has a complete copy of the repository, including the entire version history, on their local machine. • This allows users to work independently without requiring constant access to a central server. **Network Dependency:** • Users can work offline and commit changes to their local repository. • Synchronization with the central repository is required only when sharing changes with others. **Branching and Merging:** • Branching and merging are typically easier and more flexible in DVCS. • Each user can have their own branches locally, and merging is often less complex. **Speed:** • Many operations can be faster in a DVCS, as they are performed locally without relying on network access.

	History: • Each local copy contains the full history of the project, allowing users to access historical versions without needing a network connection.





Centralized Version Control System



Distributed Version Control System

Git as a Distributed Version Control System:
Git, specifically, is a widely used DVCS that offers several advantages:

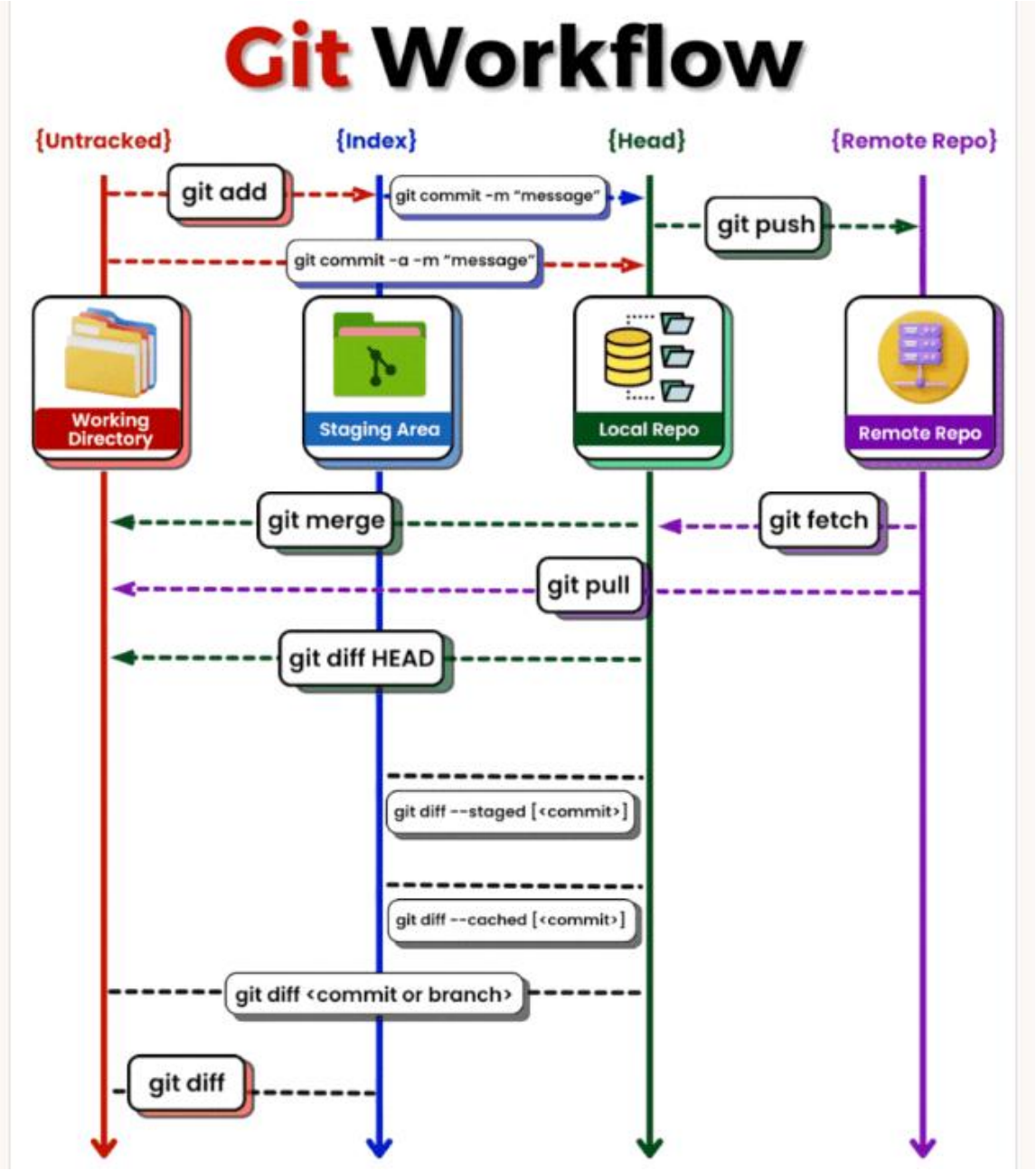
Branching and Merging:
- Git is known for its powerful and flexible branching and merging capabilities.
- Branches are lightweight, making it easy to create and switch between them.

Performance:
- Git is designed to be fast, and many operations can be performed locally without the need for a network connection.

Offline Work:
- Users can work offline, committing changes to their local repository, and then push those changes to a central repository when a network connection is available.

Full History:
- Each clone of a Git repository contains the complete history, making it robust and allowing users to work independently.

Working with Git Repositories



Repositories in Git are of two types:

- **Local Repository**:
  - Git allows the users to perform work on a project from all over the world because of its Distributive feature.
  - This can be done by cloning the content from the Central repository stored in the GitHub on the user's local machine.
  - This local copy is used to perform operations and test them on the local machine before adding them to the central repository.
- **Remote Repository**:
  - Git allows the users to sync their copy of the local repository to other repositories present over the internet.
  - This can be done to avoid performing a similar operation by multiple developers.
  - Each repository in Git can be addressed by a shortcut called remote.

Bare repositories:

- A bare repository is a remote repository that can interact with other repositories but

there is no operation performed on this repository.

- There is no Working Tree for this repository because of the same.
- A bare repository in Git can be created on the local machine of the user with the use of the following command:
    - # git init --bare
- A bare repository is always created with a .git extension.
- This is used to store all the changes, commits, refs, etc. that are being performed on the repository. It is usually a hidden directory.
- A Git repository can also be converted to a bare repository but that is more of a manual process. Git doesn't officially provide the support to do the same.

Renaming a Remote Repository:

git remote rename <old-repo-name> <new-repo-name>

Git VS GitHub

S.No.	Git	GitHub
1	Git is a software.	GitHub is a service.
2	Git is a command-line tool	GitHub is a graphical user interface
3	Git is installed locally on the system	GitHub is hosted on the web
4	Git is maintained by linux.	GitHub is maintained by Microsoft.
5	Git is focused on version control and code sharing.	GitHub is focused on centralized source code hosting.
6	Git is a version control system to manage source code history.	GitHub is a hosting service for Git repositories.
7	Git was first released in 2005.	GitHub was launched in 2008.
8	Git has no user management feature.	GitHub has a built-in user management feature.
9	Git is open-source licensed.	GitHub includes a free-tier and pay-for-use tier.
10	Git has minimal external tool configuration.	GitHub has an active marketplace for tool integration.
11	Git provides a Desktop interface named Git Gui.	GitHub provides a Desktop interface named GitHub Desktop.
12	Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc.	GitHub competes with GitLab, Bit Bucket, AWS Code Commit, etc.

Start from: https://www.geeksforgeeks.org/working-on-git-bash/?ref=lbp

1. Introduction to Git:

- Git is a Version Control System (VCS).
- VCS meaning, it can contain several versions of a file.
- VCS helps in tracking of files.
- Recovery of files is easy.
- "Issues" can be introduced and also who introduced it, could be tracked.
- Rollback of code to previous state.
- It effectively tracks changes to source code, enabling effortless branching, merging, and versioning.

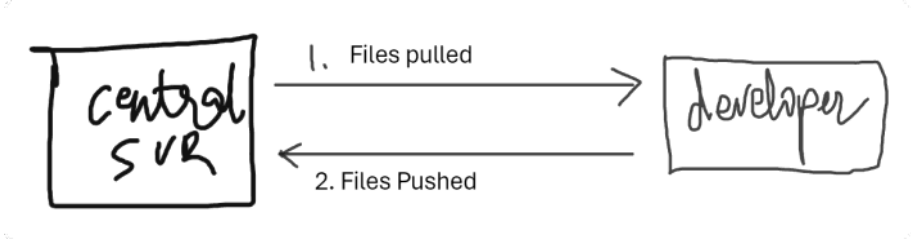
Starting with

Local VCS:

- Pros: can track files & rollback.
- Cons: if HDD is crashed/lost/damaged, everything is LOST.

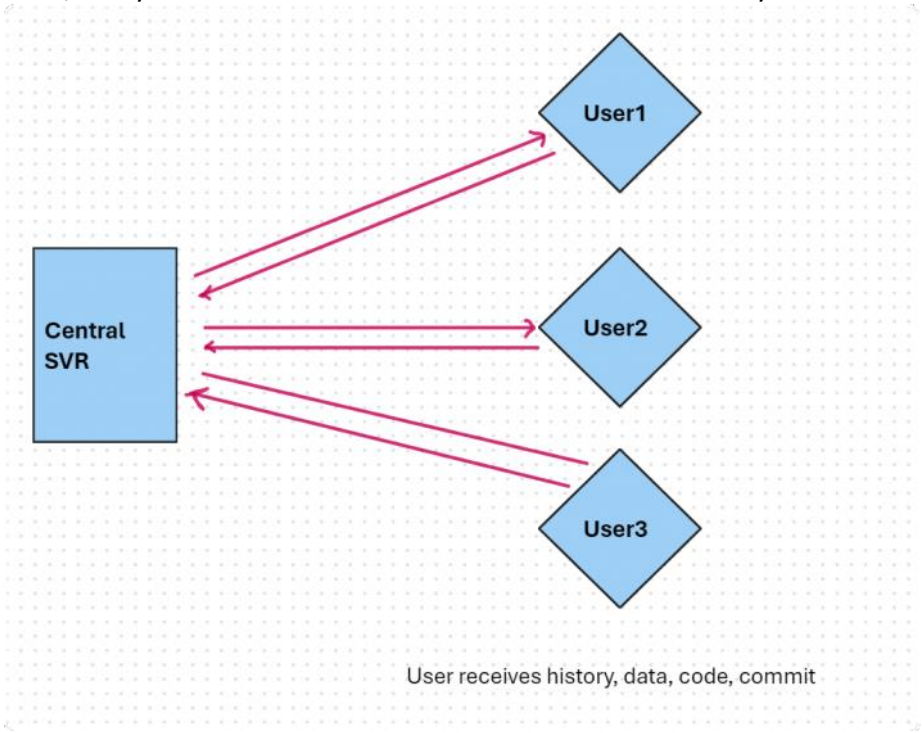
Centralized VCS:

- Strong the code to a central location.
- Resolves issues of local VCS.
-



Distributed VCS:

- Here, every user will have his own full-set of data with history and commits.



Working with Git:

- When a folder is initialized with Git, it becomes a repository—a special location where Git logs all changes made to a hidden folder.

- In that folder, each time you change, add, or remove a file, Git takes note of the change and marks the file as “modified.”
- You can choose which modified files you want to save by staging them, so don’t worry.
- Consider staging as getting the changes ready for a particular snapshot that you want to keep.
- Once the staged changes are to your satisfaction, commit them, and Git will keep a permanent copy of those files in its history.
- Git is great because it maintains a complete record of each commit you make, allowing you to see.

What is GitHub?

- GitHub is a hosting service for Git repositories and if you have a project hosted on GitHub.
- GitHub allows you to store your repo on their platform.
- It is also comes with GitHub, ability to collaborate with other developers from any location.
- It's developed by Linus B. Torwalds.

Developers can review project history to find out:

- Which changes were made?
- Who made the changes?
- When were the changes made?
- Why were changes needed?

Characteristics of Git:

- Faster
- Reliable
- Captures snapshots
- Contains all history in a single file (.git)
- Almost every operation is locally executed.
- Git has integrity.
- Git generally adds the data.

Git Installation:

- Git Command Line Tool.
- Git bash (terminal program).

2. Git - Environment Setup & init commands

Installation:
Step 1. Go to git-scm.com/downloads.
Step 2. If you are using Linux distribution: sudo apt-get install git-core
Step 3. enter “git --version”.

Setting Git Environment:
Setting username:
git config --global user.name "Jitendra"

Setting email id:
git config --global user.email "jitendrastomar5593@gmail.com"

To set the editor:
git config --global core.editor vim

List of Git config:
git config --list

To list username:
git config user.name

To list email address:
git config user.email

Initializing a local repository:
git init

Checking status of the repository:
git status

Adding files to the repository:
git add

Committing changes:
git commit -m "Your commit message"

Accessing log of commits:
git log

Git clone command:
git clone "Remote_repo_URL"
git clone "https://github.com/jitendrastomar5593/PersonalDO.git"

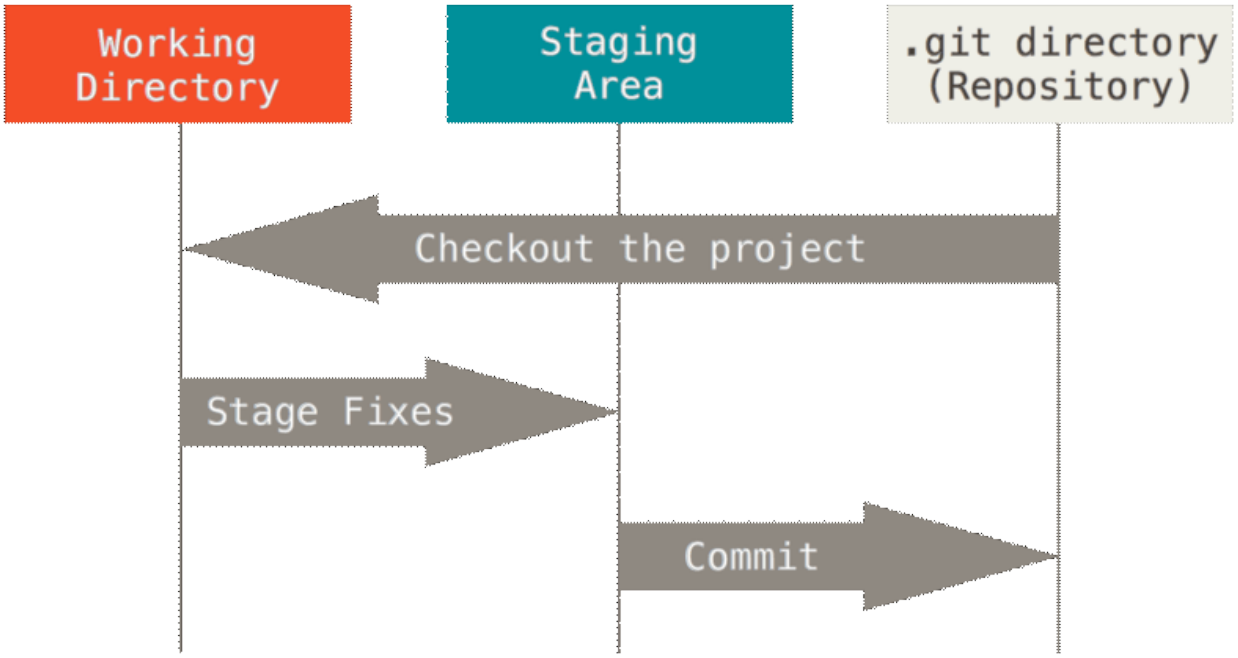
Parallel development commands
This command allows to create a branch (exact copy) for the project.
git branch branch_name

This command allows to switch from one branch to another.
git checkout branch_name

This command allows to merge a code of 2 branches in one branch.
git merge branch_name

3. Three stage architecture of Git

Sunday, November 26, 2023 9:33 PM



Working directory: your local computer working folder.
Staging area: here we keep files that are needed to be committed to final repo.
Git Directory: it's the actual repo. Container ".git" file in it.

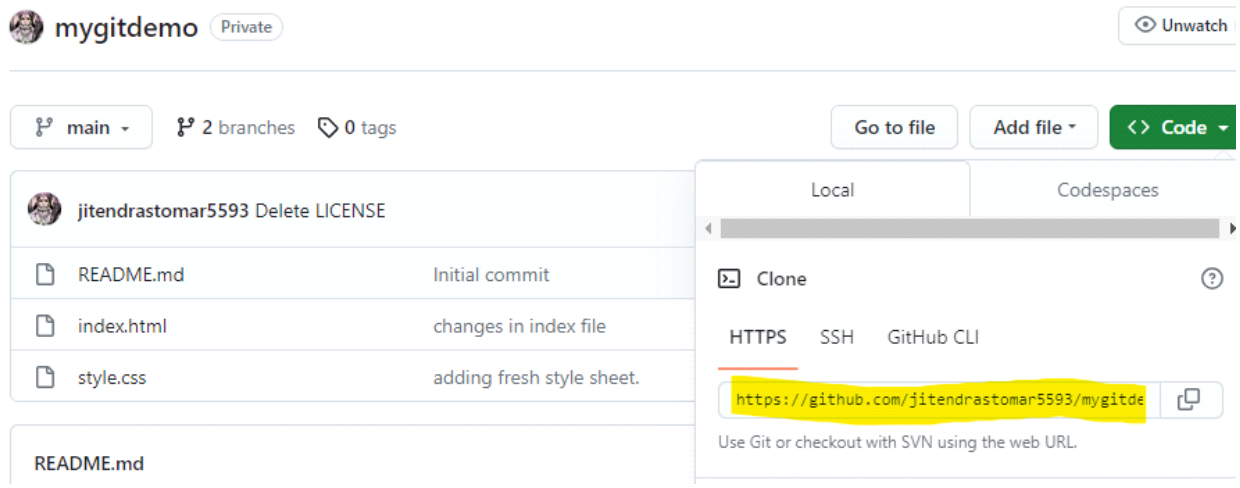
The basic Git workflow goes something like this:

1. You modify files in your working tree.
2. You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

4. Cloning a remote repo from GitHub:

Sunday, November 26, 2023 9:46 PM

Go to GitHub which you want to clone & copy the URL:

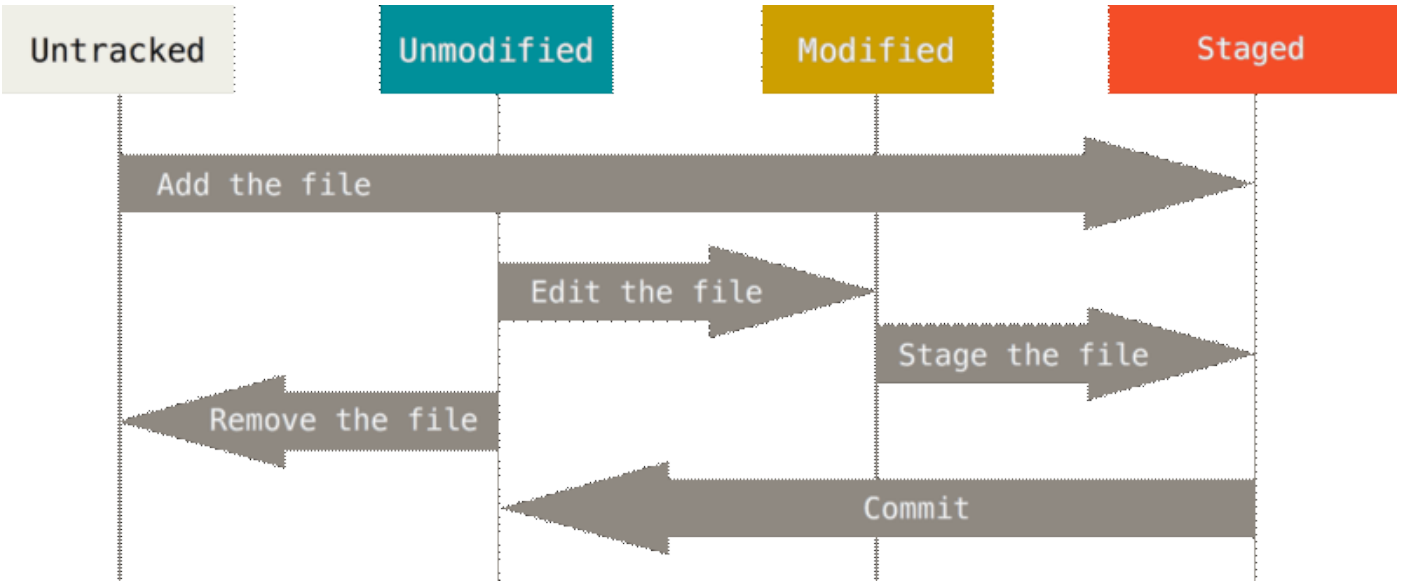


To create a clone:
git clone <URL>
git clone https://github.com/jitendrastomar5593/mygitdemo.git

To rename the repo name, while cloning:
git clone <URL> <new-name>
git clone https://github.com/jitendrastomar5593/mygitdemo.git mygit

```
admin@WorkPC MINGW64 ~/git-demo
$ ls

admin@WorkPC MINGW64 ~/git-demo
$ git clone https://github.com/jitendrastomar5593/mygitdemo.git mygit
Cloning into 'mygit'...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (16/16), done.
Receiving objects: 100% (19/19), 15.79 KiB | 2.25 MiB/s, done.Receiving objects:  78% (15/19)

Resolving deltas: 100% (5/5), done.

admin@WorkPC MINGW64 ~/git-demo
$ ls
mygit/
```

5. File Status lifecycle:



- **Untracked**: in this stage, file aren't add for the staging yet. Could be checked using "git status" cmd.
- **Unmodified**: this stage could be achieved by using cmd "git add --a". This means that file are now added to the staging are with all modification required and ready to be committed.
- **Modified**: this stage come when there's a modification in a staged (by using cmd: "git add --a") file.
- **Staged**: this stage is achieved when all file are in unmodified state. If a file/directory is in this stage, it means that the file is ready to committed now.

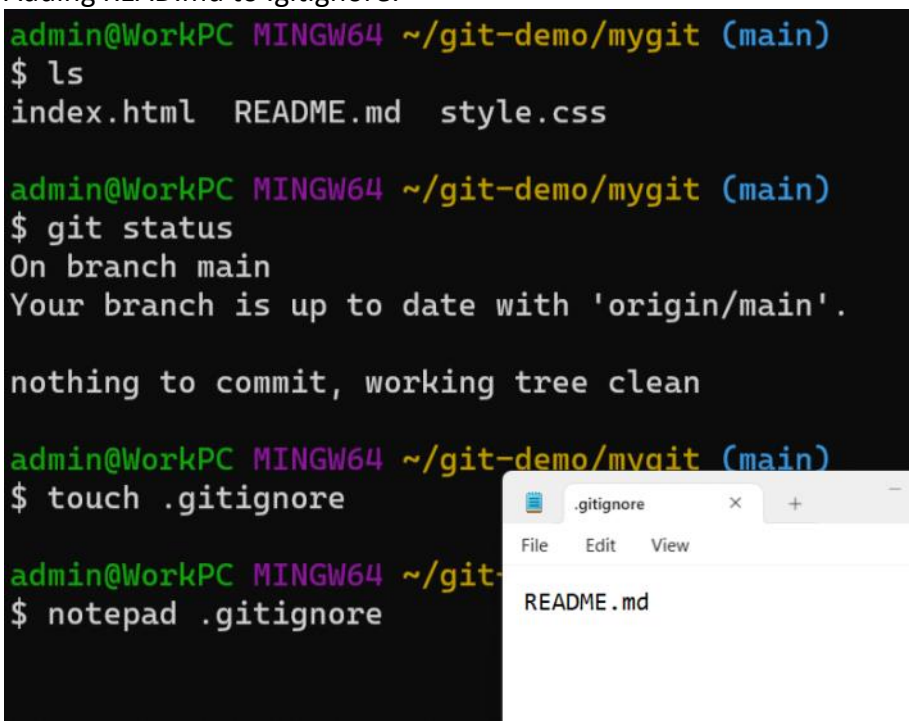
6. Git ignore file

- This file helps in ignoring all the other unwanted files in the code directory to be staged.
- This is achieved by creating a file named ".gitignore".
- All the files that must be left untracked & un-staged, must have their names this file entered manually.

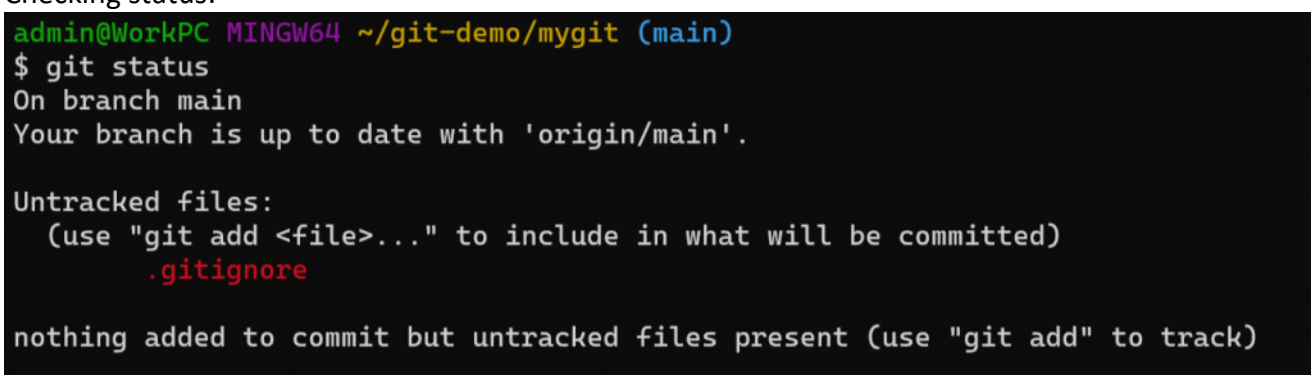
Basic commands related to .gitignore:
touch .gitignore
touch error.log // creating a file that must remain untracked.
git status // shows .gitignore & error.log as untracked.

Open .gitignore & add error.log name to it & check status again.
git status
git add .
git commit -m "ignore added"

Adding READ.md to .gitignore:



Checking status:



Adding file(s) to stage:

```
admin@WorkPC MINGW64 ~/git-demo/mygit (main)
$ git add --a

admin@WorkPC MINGW64 ~/git-demo/mygit (main)
$ git commit -m "added ignore file"
[main 356b02b] added ignore file
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore

admin@WorkPC MINGW64 ~/git-demo/mygit (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Pushing code to the repo:

```
admin@WorkPC MINGW64 ~/git-demo/mygit (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 357 bytes | 357.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jitendrastomar5593/mygitdemo.git
   a8da5eb..356b02b  main -> main
```

7. Git diff

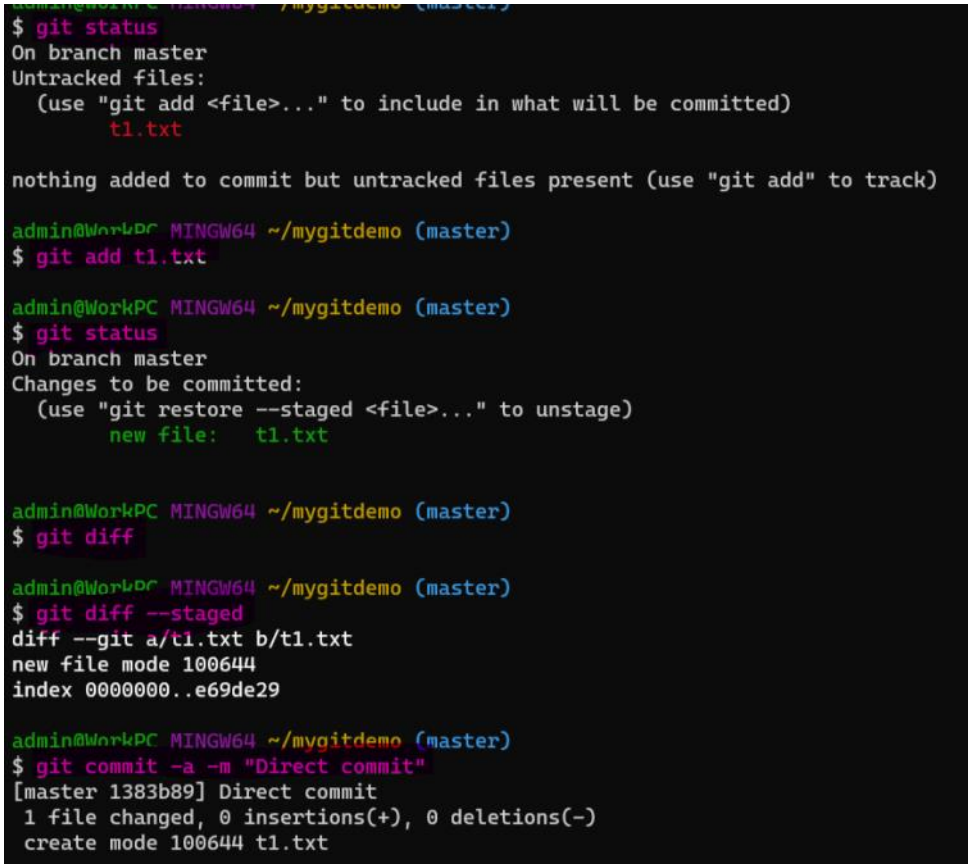
- It compares "working area" and "staging area".
- Shows, what was present earlier & what's present now.

Command:
git diff

If you want to see staging area vs old commit:
git diff --staged

Skipping the staging area:
git commit -a -m "Direct commit"

- This will commit all the 'tracked' & 'modified' files but 'will not' touch any 'untracked' files.



Renaming git files:
git mv <old-name> <new-name>
git mv t2.txt new_t2.txt //ensure that the file t2.txt is already getting tracked, else it will give fatal error.

Deleting tracked file:
rm <filename.txt>

To untrack all .gitignore files.

```
# git rm --cached <file-name>
# git status                //shows deleted file.
```

8. Git Log

To view logs
git log

```
$ git log
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

commit 20b81f7a5ff53412c274a9b9a16ec69a6e76c94e
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:20:14 2023 +0530

    t2 renamed.

commit dff80019acda6fab288673a37233506acd5ce580
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:18:23 2023 +0530

    t1 removed
```

To view log with actual values changes (line-by-line):
git log -p

```
$ git log -p
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..cde045f
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+new_t2.txt
diff --git a/new_t2.txt b/new_t2.txt
deleted file mode 100644
index e69de29..0000000
```

To view 2 recent logs:
git log -p -2

```
$ git log -p -2
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..cde045f
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+new_t2.txt
diff --git a/new_t2.txt b/new_t2.txt
deleted file mode 100644
index e69de29..0000000

commit 20b81f7a5ff53412c274a9b9a16ec69a6e76c94e
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:20:14 2023 +0530

    t2 renamed.

diff --git a/new_t2.txt b/new_t2.txt
new file mode 100644
index 0000000..e69de29
```

To view precise logs:
git log --stat

```
$ git log --stat
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

 .gitignore | 1
 new_t2.txt | 0
 2 files changed, 1 insertion(+)
```

To view logs with hashes in one line:
git log --pretty=oneline

```
$ git log --pretty=oneline
07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master) added .gitignore
20b81f7a5ff53412c274a9b9a16ec69a6e76c94e t2 renamed.
dff80019acda6fab288673a37233506acd5ce580 t1 removed
1383b8928f2a6c9f394dfd46f3c3a25d39273e75 Direct commit
c4028500a12404586737b3653e8be4de6aa407d6 added context to README.md and removed Lic file.
ff5c3008bb8749ecca3e6cdba4ad8451a71298c8 (origin/master) changes in index file
325ff31d988c59978d711261f761600855ab38f4 Merge pull request #2 from jitendrastomar5593/master
a2d162053b98b8aec9ff6b739cf8e38fa2e744a9 adding fresh style sheet.
ea3693279f4a9e801eff4319d0fcc3bf37dbe418 Merge pull request #1 from jitendrastomar5593/master
0034840d74891baa12315445c8ad21f5ad90651a index file done
5c4306005572497a37e66cbe8f8892649dc71d3d Initial commit
```

To view logs in short:
git log --pretty=short

```
$ git log --pretty=short
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>

    added .gitignore

commit 20b81f7a5ff53412c274a9b9a16ec69a6e76c94e
Author: Jeetu <jitendrastomar5593@gmail.com>

    t2 renamed.

commit dff80019acda6fab288673a37233506acd5ce580
Author: Jeetu <jitendrastomar5593@gmail.com>

    t1 removed
```

To view logs with full details:
git log --pretty=full

```
$ git log --pretty=full
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Commit: Jeetu <jitendrastomar5593@gmail.com>

    added .gitignore
```

Author: person who created the file:
Commit: person who committed changed to that file.

To list logs since last 2 days:
git log --since=2.days

```
$ git log --since=2.days
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

commit 20b81f7a5ff53412c274a9b9a16ec69a6e76c94e
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:20:14 2023 +0530

    t2 renamed.

commit dff80019acda6fab288673a37233506acd5ce580
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:18:23 2023 +0530

    t1 removed
```

git log --since=2.weeks
git log --since=2.months
git log --since=2.years

To list hash, commit with author name:
git log --pretty=format:"%h -- %an"

```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git log --pretty=format:"%h -- %an"
07828e7 -- Jeetu
20b81f7 -- Jeetu
dff8001 -- Jeetu
1383b89 -- Jeetu
c402850 -- Jeetu
ff5c300 -- Jeetu
325ff31 -- Jitendra Singh Tomar
a2d1620 -- BOB
ea36932 -- Jitendra Singh Tomar
0034840 -- Alice
5c43060 -- Jitendra Singh Tomar

admin@WorkPC MINGW64 ~/mygitdemo (master)
$
```

To list hash, commit with author email:
git log --pretty=format:"%h -- %ae"

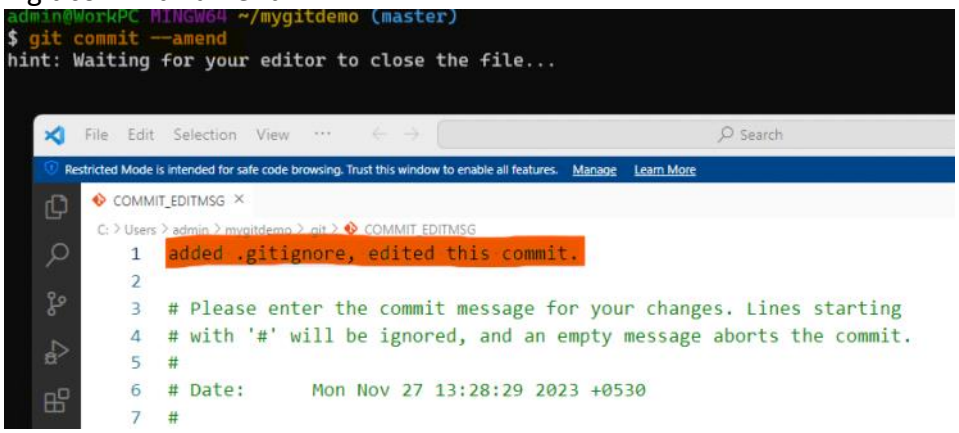
```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git log --pretty=format:"%h -- %ae"
07828e7 -- jitendrastomar5593@gmail.com
20b81f7 -- jitendrastomar5593@gmail.com
dff8001 -- jitendrastomar5593@gmail.com
1383b89 -- jitendrastomar5593@gmail.com
c402850 -- jitendrastomar5593@gmail.com
ff5c300 -- jitendrastomar5593@gmail.com
325ff31 -- jitendrastomar5593@gmail.com
a2d1620 -- bob@jeetusingh.in
ea36932 -- jitendrastomar5593@gmail.com
0034840 -- alice@jeetusingh.in
5c43060 -- jitendrastomar5593@gmail.com
```

Editing the commit for an existing (last) commit:
git log -p -1

```
$ git log -p -1
commit 07828e77a4e2be1fdd74647ae517ffa0e562a470 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 13:28:29 2023 +0530

    added .gitignore

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..cde045f
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+new_t2.txt
diff --git a/new_t2.txt b/new_t2.txt
deleted file mode 100644
index e69de29..0000000
```

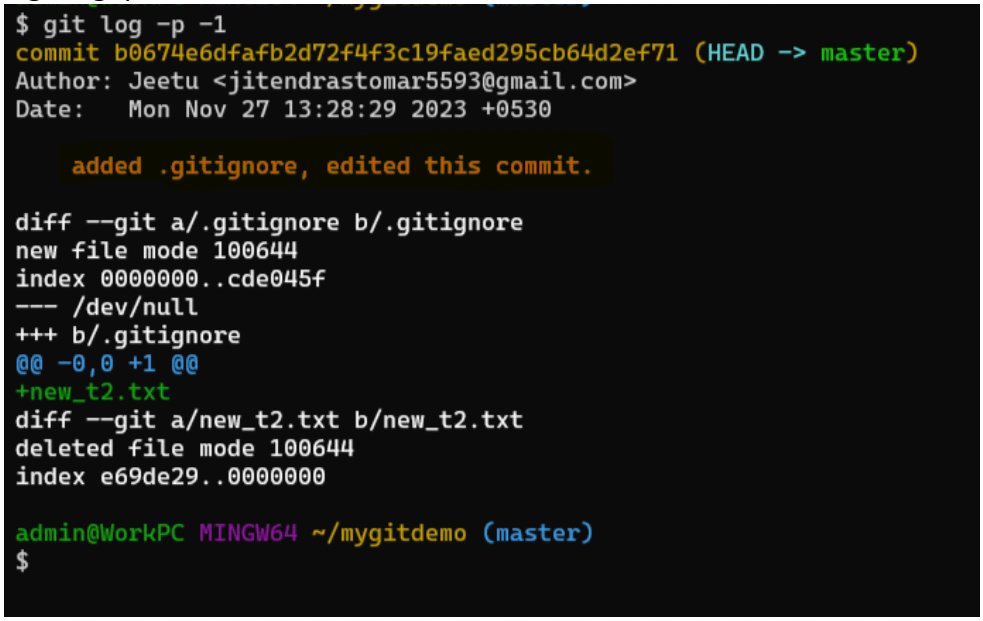
Editing commit

git commit --amend



Verifying the edited commit:

git log -p -1

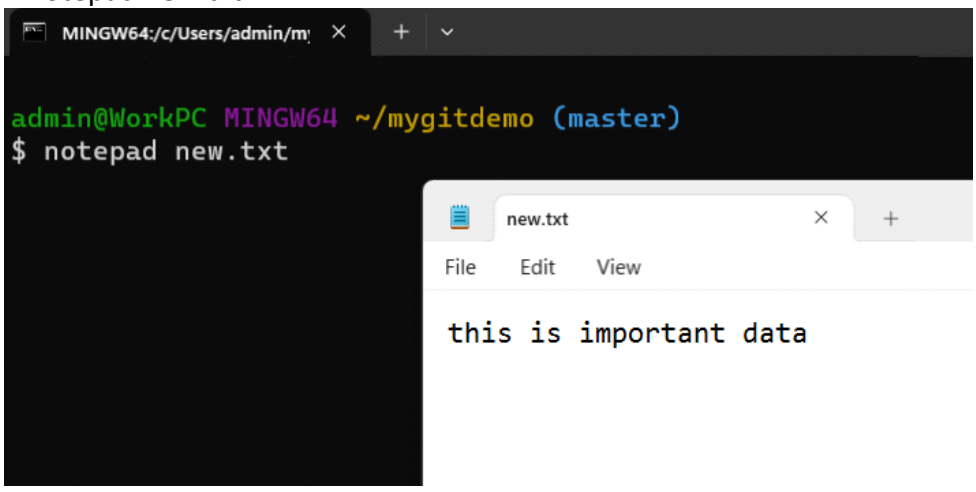


9. Un-staging & un-modifying in Git:

To un-stage a file (its only done on staged files, not on committed files):
git restore --staged <file-name.txt>

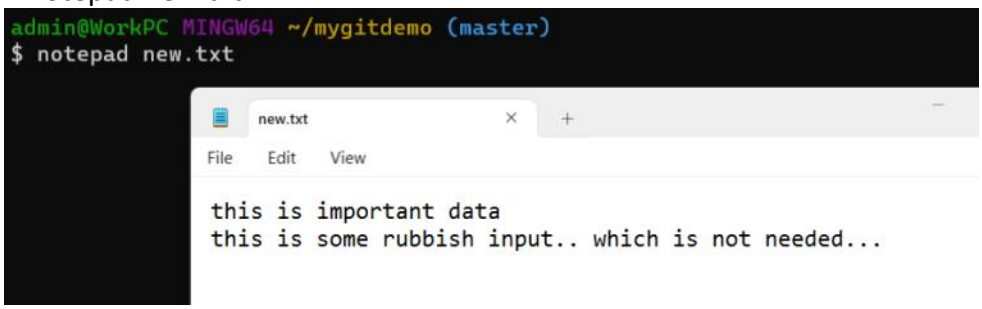
```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ touch t3.txt

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git add t3.txt

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   t3.txt


admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git restore --staged t3.txt

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        t3.txt

nothing added to commit but untracked files present (use "git add" to track)

admin@WorkPC MINGW64 ~/mygitdemo (master)
$
```

Restoring/Revert/Unmodified file:
notepad new.txt

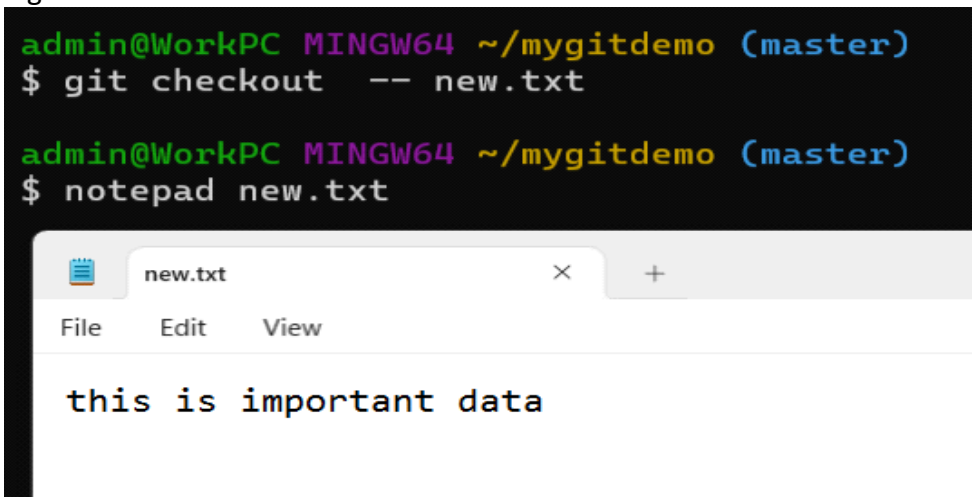


```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new.txt

nothing added to commit but untracked files present (use "git add" to track)

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git add new.txt

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git commit -m "performing revert"
[master 5babc48] performing revert
 1 file changed, 1 insertion(+)
 create mode 100644 new.txt

admin@WorkPC MINGW64 ~/mygitdemo (master)
$
```

Re-writing same new.txt file:
notepad new.txt



Reverting to last known commit:
git checkout -- <filename.txt>



To revert/checkout multiple files:
git checkout -f
git status

10. Remote repositories:

To add a remote URL as origin (origin is just a commonly used name, it could be different as well):
git remote add origin git@github.com:<URL>
git remote add origin git@github.com:jitendrastomar5593/mygitdemo.git
git remote
git remote -v

Pushing data/code to repo:
git push -u origin master

If pushing shows error, then need to generate SSH keys and attach it to GitHub.com account:
ssh-keygen -t rsa -b 4096 -c "jitendrastomar5593@gmail.com"

Adding SSH key to the agent:
ssh-add ~/.ssh/id_rsa

Fetch the public ssh key & add it to github:
GitHub.com --> settings --> SSH & GPG keys --> save the public key.

Now run the command:
git push -u origin master // now it should work.

11. Git Alias

Creating alias

git config --global alias.st status

```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git config --global alias.st status

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git st
On branch master
nothing to commit, working tree clean

admin@WorkPC MINGW64 ~/mygitdemo (master)
$
```

git config --global alias.ci commit

```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git config --global alias.ci commit

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git ci
On branch master
nothing to commit, working tree clean
```

git config --global alias.last 'log -p -1'

```
admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git config --global alias.last 'log -p -1'

admin@WorkPC MINGW64 ~/mygitdemo (master)
$ git last
commit 5babc484dac96fc672edc3328cba39d0c6821be8 (HEAD -> master)
Author: Jeetu <jitendrastomar5593@gmail.com>
Date:   Mon Nov 27 14:24:43 2023 +0530

    performing revert

diff --git a/new.txt b/new.txt
new file mode 100644
index 0000000..07a2f3a
--- /dev/null
+++ b/new.txt
@@ -0,0 +1 @@
+this is important data
\ No newline at end of file
```

1. Introduction to K8s

Friday, May 19, 2023 6:35 PM

- Kubernetes is an open-source container orchestration platform that helps
  - Automate the deployment,
  - Scaling and,
  - Management of containerized applications.
- It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).
- Kubernetes provides a powerful framework for
  - managing containerized workloads and services,
  - allowing organizations to build scalable and resilient application architectures.
- At its core:
  - It allows you to define your application as a set of declarative configurations, specifying the desired state of your application and its components.
  - Kubernetes then takes responsibility for managing and maintaining that desired state, ensuring your application runs consistently and efficiently.
- The main building block of Kubernetes is the container.
  - Containers are lightweight, isolated execution environments that package an application and its dependencies.
- Kubernetes allows you to deploy these containers into a cluster of machines, called nodes.
  - These nodes can be physical machines or virtual machines running in a cloud environment.

2. Kubernetes Feature

Friday, May 19, 2023 6:36 PM

Kubernetes provides a rich set of features and functionalities, including:

- **Container Orchestration**: Kubernetes automatically schedules containers on different nodes based on resource availability and constraints, ensuring optimal resource utilization.
- **Scaling and Load Balancing**: Kubernetes enables horizontal scaling by automatically replicating containers based on defined metrics and distributing traffic across them using load balancing techniques.
- **Service Discovery and Load Balancing**: Kubernetes provides a built-in service discovery mechanism that allows containers to find and communicate with each other using DNS or environment variables. Load balancing is also handled transparently by Kubernetes.
- **Self-Healing**: Kubernetes monitors the health of containers and automatically restarts failed containers or replaces them with new ones. It also provides advanced health checks and recovery mechanisms.
- **Rolling Updates and Rollbacks**: Kubernetes supports seamless updates of applications by gradually rolling out new versions while ensuring high availability. In case of issues, it also allows for easy rollbacks to a previous known state.
- **Storage Orchestration**: Kubernetes provides mechanisms to manage storage for containers, including persistent volumes and volume claims. This allows you to store and retrieve data from within your containers.
- **Secrets and Configuration Management**: Kubernetes offers built-in support for securely managing sensitive information such as passwords, API keys, and TLS certificates. It also provides mechanisms for managing configuration data for your applications.

3. Where K8s is useful?

Friday, May 19, 2023 6:38 PM

Kubernetes is useful in a variety of scenarios and environments. Here are some common use cases where Kubernetes shines:

- **Containerized Application Deployment**:
  - Kubernetes is primarily designed for deploying and managing containerized applications.
  - If you have an application that is packaged in containers, whether it's a monolithic application (components of an application are tightly coupled and packaged together as a single unit.) or a microservices-based architecture (collection of small, loosely coupled, and independently deployable services), Kubernetes provides an excellent platform for orchestrating and scaling those containers across a cluster of nodes.

- **Scalability and High Availability**:
  - Kubernetes excels in scenarios where scalability and high availability are crucial.
  - It can automatically scale your application by adding or removing container replicas based on resource usage or other defined metrics.
  - It ensures that your application remains highly available even in the event of container or node failures.

- **Microservices Architecture**:
  - Kubernetes is well-suited for managing microservices-based architectures.
  - It allows you to deploy each microservice as a separate container and provides mechanisms for service discovery, load balancing, and communication between microservices.
  - Kubernetes also enables independent scaling and versioning of individual microservices.

- **Multi-Cloud and Hybrid Cloud Environments**:
  - Kubernetes is cloud-agnostic and can run on various cloud providers such as AWS, Google Cloud, Azure, and on-premises data centers.
  - This makes it suitable for organizations that operate in multi-cloud or hybrid cloud environments, allowing consistent deployment and management of applications across different infrastructure platforms.

- **Development and Testing Environments**:
  - Kubernetes provides a consistent environment for development and testing teams.
  - Developers can run their applications locally in containers using tools like Minikube or Docker Desktop Kubernetes, which mimic the Kubernetes environment.
  - This helps ensure that applications behave consistently in development, testing, and production environments.

- **CI/CD Pipelines**:
  - Kubernetes integrates well with CI/CD (Continuous Integration/Continuous Deployment) pipelines.
  - It allows for automated application deployment, rolling updates, and rollbacks.
  - Kubernetes can be seamlessly integrated with popular CI/CD tools like Jenkins, GitLab CI/CD, or Argo, enabling organizations to automate the deployment and delivery of their applications.

- **Big Data and AI/ML Workloads**:
  - Kubernetes can efficiently handle resource-intensive workloads such as Big Data processing and

AI/ML (Artificial Intelligence/Machine Learning).
    - By leveraging Kubernetes' ability to scale containers and distribute workloads across a cluster, organizations can run data-intensive applications and leverage distributed processing frameworks like Apache Spark or TensorFlow.

- **Internet of Things (IoT) Applications**:
    - Kubernetes can be utilized in IoT scenarios to manage and deploy containerized applications on edge devices or IoT gateways.
    - It allows organizations to deploy and manage containers on resource-constrained devices efficiently, ensuring consistency and scalability in IoT deployments.

4. Kubernetes Benefits

Friday, May 19, 2023 6:39 PM

- **Scalability and Elasticity**:
  - Kubernetes enables horizontal scaling of applications by automatically distributing container replicas across multiple nodes.
  - It can scale up or down based on workload demands, ensuring efficient resource utilization and the ability to handle increased traffic or processing requirements.

- **High Availability and Fault Tolerance**:
  - Kubernetes enhances application reliability by automatically monitoring and managing container health.
  - It restarts or replaces failed containers, reschedules them on healthy nodes, and maintains the desired state of the application. This ensures high availability and minimizes downtime.

- **Automated Deployments and Rollbacks**:
  - Kubernetes simplifies the deployment process by allowing you to define your application's desired state in declarative configurations.
  - It automates the deployment of containers, rolling updates, and rollbacks, reducing manual effort and minimizing the risk of errors during deployments.

- **Container Orchestration and Management**:
  - Kubernetes abstracts away the complexities of managing containerized applications.
  - It handles container scheduling, resource allocation, networking, and service discovery, allowing developers to focus on application development rather than infrastructure management.

- **Service Discovery and Load Balancing**:
  - Kubernetes provides built-in service discovery mechanisms that allow containers to easily find and communicate with each other.
  - It also offers load balancing for distributing traffic across containers, ensuring efficient utilization of resources and optimizing application performance.

- **Storage Orchestration**:
  - Kubernetes offers flexible and scalable storage options for containers.
  - It provides mechanisms for managing persistent volumes, allowing data to persist even if containers are terminated or rescheduled.

- **Infrastructure Flexibility**:
  - Kubernetes is cloud-agnostic and can run on various cloud platforms (such as AWS, Google Cloud, Azure) as well as on-premises infrastructure.
  - This flexibility allows organizations to adopt a multi-cloud or hybrid cloud strategy while maintaining consistency in application deployment and management.

- **Ecosystem and Community Support**:
  - Kubernetes has a vibrant ecosystem with a wide range of tools and integrations available.
  - It is supported by a large and active community of developers, which ensures continuous improvement, bug fixes, and the availability of resources and expertise.

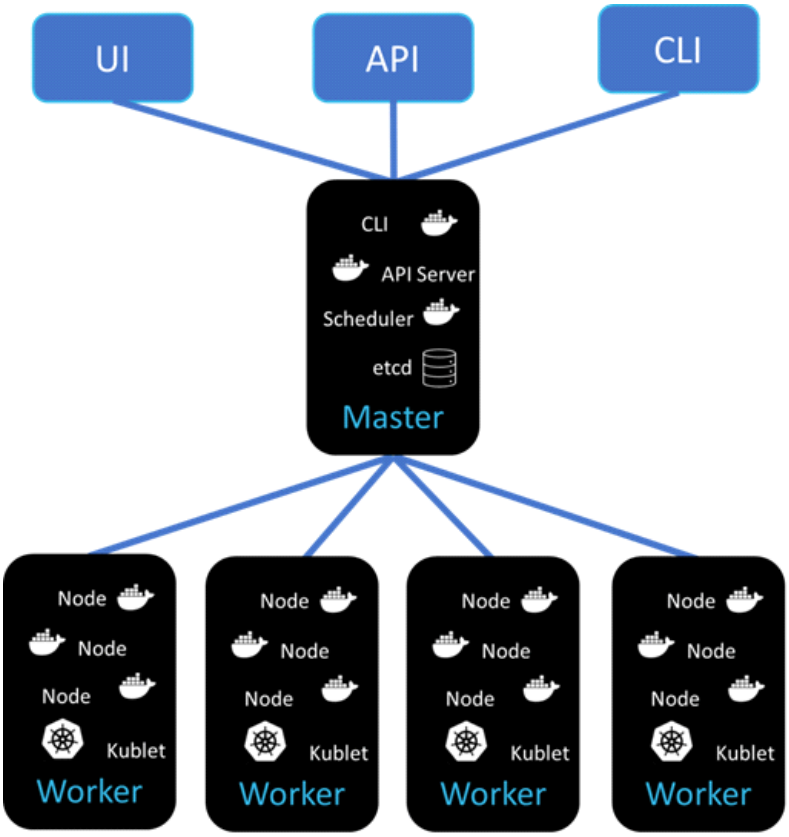
- **DevOps Enablement**:
  - Kubernetes aligns well with DevOps practices by providing the ability to automate application deployments, scaling, and management.

- It integrates with popular CI/CD tools, enabling organizations to establish efficient and automated development and deployment pipelines.

- **Cost Efficiency**:
  - Kubernetes optimizes resource utilization and allows efficient scaling, resulting in cost savings.
  - It can dynamically allocate and deallocate resources based on demand, preventing overprovisioning and reducing infrastructure costs.

5. Kubernetes Architecture

Friday, May 19, 2023 6:40 PM



A high-level K8s architecture has 4 components:

1. Master node or Control plane
2. Worker node(s)
3. Pods
4. Containers

Master Node: The master node is the control plane of the Kubernetes cluster. It coordinates and manages the cluster's operations. The master node typically runs the following components:

- **API Server**:
  - The API server exposes the Kubernetes API, which allows users and other components to interact with the cluster.
  - It serves as the central control point for managing the cluster's resources and receiving and processing requests.
- **Scheduler**:
  - The scheduler is responsible for placing containers on worker nodes based on resource availability, constraints, and scheduling policies.
  - It ensures that containers are distributed across the cluster efficiently.
- **Controller Manager**:
  - The controller manager runs various controllers that monitor the state of the cluster and ensure that the desired state is maintained.
  - These controllers handle tasks such as scaling deployments, managing replication, handling node failures, and maintaining the overall health of the cluster.
- **etcd**:
  - etcd is a distributed key-value store used by Kubernetes to store the cluster's configuration data and state.

- It provides a reliable and highly available data store for the master node.

Worker Nodes: Worker nodes, also known as minion nodes, are the machines where containers are scheduled and run.

- Runs the pods and containers (inside the pods)

  Each worker node typically runs the following components:

- **Kubelet**:
  - The Kubelet is responsible for managing containers on a node.
  - It interacts with the API server to receive instructions for creating, starting, stopping, and monitoring containers. It ensures that containers are running and healthy on the node.
- **Container Runtime**:
  - The container runtime is the software responsible for running containers, such as Docker or containerd.
  - It provides the necessary environment to execute containerized applications.
- **kube-proxy**:
  - kube-proxy is responsible for network proxying and load balancing.
  - It manages network connectivity and routing between services and containers, enabling communication within the cluster.

Networking:

- Kubernetes requires a networking solution to enable communication between containers running on different nodes.
- There are multiple networking options available, such as the Container Network Interface (CNI), which allows for different network plugins to be used, including overlay networks, software-defined networks, or bare-metal networks.

Add-ons: Kubernetes provides various add-ons and extensions that enhance its functionality. These include:

- **Dashboard**:
  - The Kubernetes Dashboard is a web-based user interface that provides a graphical representation of the cluster's resources and allows users to manage and monitor their applications.
- **Ingress Controller**:
  - An Ingress Controller manages inbound network traffic to services within the cluster and allows for the configuration of routing rules and load balancing for external access.
- **DNS**:
  - Kubernetes includes a DNS add-on that provides a DNS service within the cluster, enabling containers to communicate with each other using DNS names.
- **Metrics Server**:
  - The Metrics Server collects resource usage metrics from the cluster and makes them available to the Kubernetes API. It is used for auto-scaling and resource management.

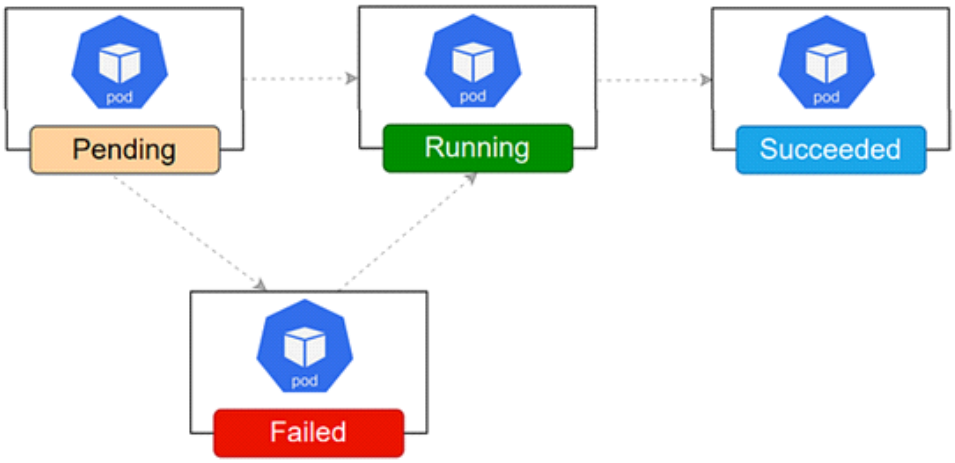
6. Pod & its life cycle

Friday, May 19, 2023 6:42 PM

To keep it simple:

- Virtualization = Virtual Machines
- Docker = Containers
- Kubernetes = Pods

- In simple terms, Pods are the workloads that runs on worker nodes
- A pod is the smallest and most fundamental unit of deployment.
- It represents a single instance of a running process in the cluster.
- A pod encapsulates one or more containers, storage resources, and network resources, and it is the basic building block of an application in Kubernetes.
- Pods play a vital role in Kubernetes, providing a logical unit for managing containers and enabling applications to be scheduled and deployed as a cohesive unit.
- They enable efficient resource sharing, easy inter-container communication, and encapsulation of application components within a single entity.



1. When a pod is created, it goes in PENDING state.
2. PENDING state means, your pod is accepted for the deployment but hasn't scheduled onto any of the VMs (nodes).
3. Schedular check for resources like CPU & RAM requirements and puts it into a specific VM within cluster.
4. Now POD status changes from PENDING to CREATING.
5. In CREATING state, image is getting pulled from centralized repository (in our case Docker Hub). If the image is already present locally, this pulling will be skipped.
6. Once image is pulled, container status changes from CREATING to RUNNING state.
7. If it fails to pull the image due to any reason, it will change to FAILED state.
8. RUNNING state means, now the program/application is running properly.
9. Now in case, if a service within fails or crashes due to any reason, schedular will restart the pod.
10. But if it crashes frequently, the container state changes to "CRASH LOOP BACK OFF" state. Here the pod tries to heal itself. But if it fails to do so, you need to start looking into commands like "kubectl get pods" or "kubectl describe pod <pod-name>" or even need to check logs.

Why Pods???

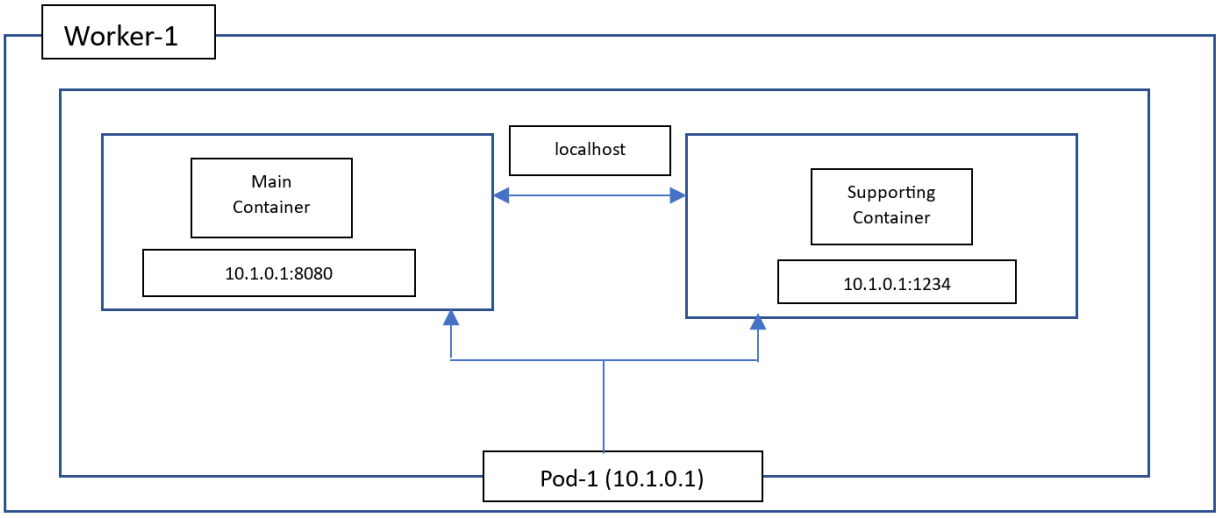
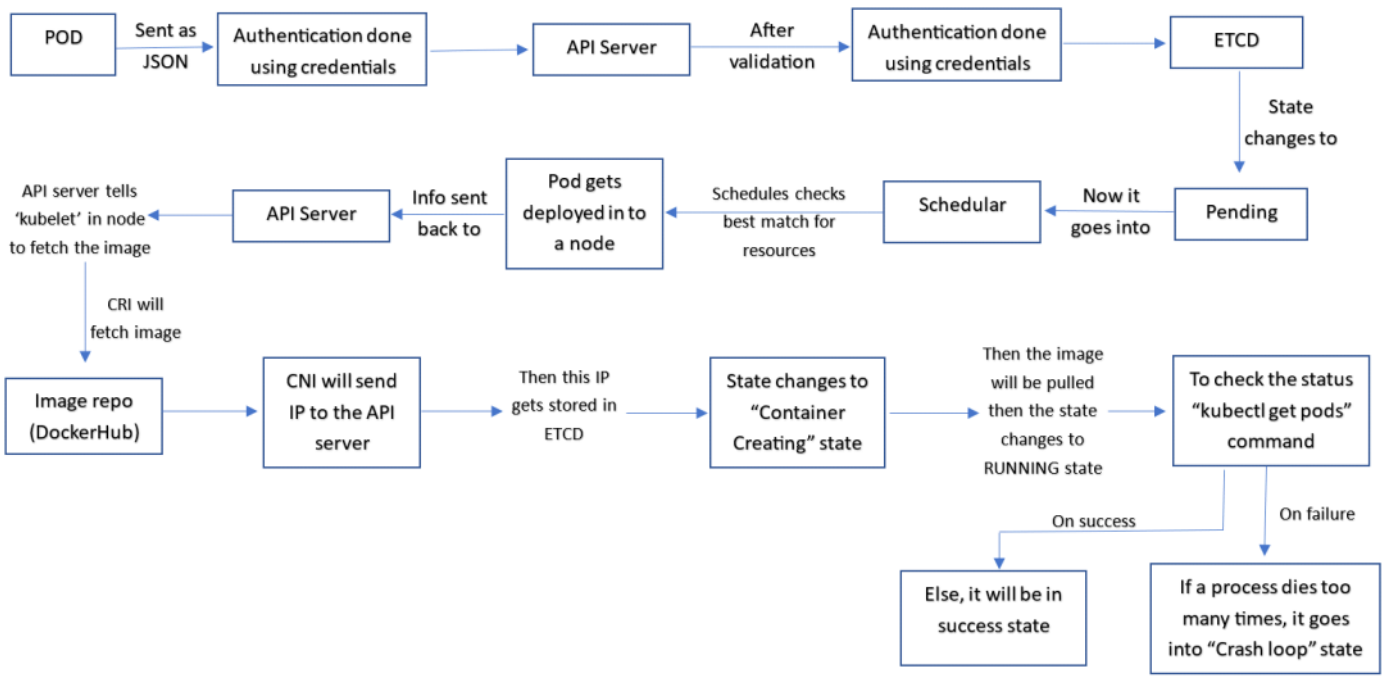


Fig: multi-container pod

- Each container will have their own port number.
- To access main container, use: **10.1.0.1:8080**
- To access supporting container, use: **10.1.0.1:1234**
- To access supporting container from main container, use: **localhost:1234**

7. Pod life cycle - Detailed

Wednesday, May 24, 2023 9:09 AM



this happens on **MASTER node**

1. pod created --> pending state --> deployment is accepted, but hasn't scheduled on any node.
2. "schedular" will check for the resources (CPU, Mem) & then it will be pushed to that worker node.

this happens on **WORKER node**

3. pod status changes from PENDING to CREATING
4. in this state, image will be checked (locally), else it will download the image from DockerHub.
5. once, image is pulled, container state changed from CREATING to RUNNING.
6. if it fails due to any reason, the state will change to FAILED.
7. RUNNING state means that the application is deployed.
8. if the container fails, K8 will restart the container/pod.
9. if the crash happens frequently, then state changes to "CRASHLOOP BACK OFF"
10. then pod tries to HEAL itself. if success -> OK, else - TS manually.

    #kubectl get pods
    #kubectl describe pod <pod-name>
    #check logs

8.1 Installing K8s on cloud

Wednesday, May 24, 2023 10:20 PM

0. Provisioning Nodes and Firewalls:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

0a. Kubernetes Cluster Nodes(3):
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Cloud: Google Compute Engine (GCE)
Master(1): 2 vCPUs - 4GB Ram
Worker(2): 2 vCPUs - 2GB RAM
OS: Ubuntu 18.04 or CentOS/RHEL 7

0b. Firewall Rules (Ingress):
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Master Node: 2379,6443,10250,10251,10252
Worker Node: 10250,30000-32767

0c. NOT Mandatory. For better visibility.

Add below lines to ~/.bashrc
Master Node:
PS1="\e[0;33m[\u@\h \W]\$ \e[m "

Worker Node:
PS1="\e[0;36m[\u@\h \W]\$ \e[m "

****

1. PRE-Reqs: Disable Swap | Bridge Traffic (Run it on MASTER & WORKER Nodes):
~~~~~~

1a) **Disable SWAP:**
~~~~~~~~~~~~~~~~~~

swapoff -a
sed -i.bak -r 's/(.+ swap .+)/#\1/' /etc/fstab

1b) **Bridge Traffic:**
~~~~~~~~~~~~~~~~~~~~

lsmod | grep br_netfilter
sudo modprobe br_netfilter
lsmod | grep br_netfilter

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1

```
EOF

sudo sysctl --system
```

****

2. Installing Docker (Run it on MASTER & WORKER Nodes):

~~~~~~

```
apt-get update -y
apt-get install -y  apt-transport-https ca-certificates curl software-properties-common gnupg2

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"
```

2a) Installing Docker:

~~~~~~~~~~~~~~~~~~~~~~

```
apt-get update && sudo apt-get install -y \
  containerd.io=1.2.13-2 \
  docker-ce=5:19.03.11~3-0~ubuntu-$(lsb_release -cs) \
  docker-ce-cli=5:19.03.11~3-0~ubuntu-$(lsb_release -cs)

############################################################
apt-get update && sudo apt-get install -y \
  containerd.io \
  docker-ce \
  docker-ce-cli
############################################################
```

2b) Setting up the Docker "daemon":

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

mkdir -p /etc/systemd/system/docker.service.d
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

2c) Start and enable docker:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

systemctl daemon-reload
systemctl enable docker
systemctl restart docker
systemctl status docker

****

3. Installing KUBEADM - KUBELET - KUBECTL (On Master & Worker node)
~~~~~~

apt-get update -y && 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

3a) Installing Kubeadm, Kubelet, Kubectl: (On Master & Worker node)
~~~~~~

apt-get update -y && apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

3b) Start and enable Kubelet: (On Master & Worker node)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

systemctl daemon-reload
systemctl enable kubelet
systemctl restart kubelet
systemctl status kubelet

****

4. Initializing CONTROL-PLANE (Run it on MASTER Node only)
~~~~~~

kubeadm init

##################################
If this gives error "[ERROR CRI]: container runtime is not running:"
execute below commands:
    rm /etc/containerd/config.toml
    systemctl restart containerd
    kubeadm init
~~~~~~

##################################

****

5. Installing POD-NETWORK add-on (Run it on MASTER Node only)
~~~~~~

5a) "kubectl":
~~~~~~~~~~~~~~~

```
# for kubectl
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

5b) Joining the workers to the master:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
kubeadm join 10.190.0.2:6443 --token u81ks2.84wwdap851fqghd0 \
      --discovery-token-ca-cert-hash
sha256:a2f38616d810bfe21c4499b359dd7921b4faea8d5422b4f7d461925668fab503
```

##################################
If this gives error "[ERROR CRI]"
execute below commands:
        rm /etc/containerd/config.toml
        systemctl restart containerd
        kubeadm join ...
##################################

to check the status, if its connected or not (On master node):
kubectl get nodes

STATUS = NotReady, due to absense of Pod-Network (step: 5c)

5c) Installing "Weave CNI" (Pod-Network add-on):
~~~~~~

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

site command (working): kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

NOTE: There are multiple CNI Plug-ins available. You can install choice of yours. Incase above commands doesn't work, try checking below link for more info.
Check URL for weave-net: https://www.weave.works/docs/net/latest/kubernetes/kube-addon/

to check the status, if its connected or not (On master node):
kubectl get nodes

STATUS = Ready (should be)

****

6. Joining Worker Nodes (Run it on WORKER Node only):
~~~~~~

Past the Join command from above kubeadm init output
kubeadm join <...>

Run this command IF you do not have above join command and/or to create NEW one.
kubeadm token create --print-join-command

****
~~~~~~

8.2 Installing K8s on-prem

Sunday, May 28, 2023 8:48 PM

1. Turn off SELinux, Firewalld

2. Master node should have static IP.

3. update your repos (master, workers)
   # apt-get update -y

4. turn off/disable SWAP (master, workers) -- IMPORTANT STEP
   # swapoff -a
   # vim /etc/fstab

5. install openssh (master, workers)
   # apt-get install -y openssh-server
   # apt-get update -y

6. adding the kube repo
   # apt-get install -y apt-transport-https curl
   # curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
   # cat <<EOF > /etc/apt/sources.list.d/kubernetes.list
   > deb http://apt.kubernetes.io kubernetes-xenial main
   > EOF
   # apt-get update -y

7. install docker, kubelet, kubectl, kubeadm (master, workers)
   A . set up docker

##########

   1. sudo apt-get update -y
   2. sudo apt-get install ca-certificates curl gnupg -y
   3. sudo install -m 0755 -d /etc/apt/keyrings
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
   sudo chmod a+r /etc/apt/keyrings/docker.gpg
   4. echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

5. sudo apt-get update -y
6. sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

##

8. install kube commands:
   # apt-get install -y kubelet kubectl kubeadm

9. edit the config file:
   # vim /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
   & add this to the bottom of the file
   Environment="cgroup-drive=systemd/cgroup-driver=cgroupfs"
   :wq!

~~~~~~

On master node:
---

10. Initialize k8s
    # kubeadm init

11. Create kube directory
    # mkdir -p $HOME/.kube

12. Copy config file to this dir.
    # sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

13. Provide the permissions
    # sudo chown $(id -u):$(id -g) $HOME/.kube/config

14. Check the nodes status
~~~~~~

kubectl get nodes

15. List node information in detail.
    # kubectl get pods -o wide --all-namespace

On worker nodes:
---
16. join the cluster (workers) - using the output given by init command
    # kubeadm join 192.168.88.128:6443 --token 85yi35.nw2yti1uvsezqt3p \
    --discovery-token-ca-cert-hash sha256:5cba15cef79dfef9b2427c3cae38d485a75b58875a6d6c76be4643e38fbc48c3

9. Kubernetes network add-ons

Wednesday, May 24, 2023 10:35 PM

Link: https://kubernetes.io/docs/concepts/cluster-administration/addons/

There are several networking plugins available for Kubernetes.
Here is a comprehensive list of networking plugins commonly used in Kubernetes:

- Flannel
- Calico
- Weave (https://www.weave.works/docs/net/latest/kubernetes/kube-addon/)
- Cilium
- Antrea
- Canal (Combination of Calico and Flannel)
- kube-router
- Romana
- Kube-OVN
- Contiv
- Project Calico with BGP
- kube-iptables-tailer
- Multus CNI
- SR-IOV
- Macvlan
- IPvlan
- OpenContrail
- Gobetween
- Nuage Networks
- Kuryr
- NSX-T
- Azure CNI
- AWS VPC CNI
- GKE VPC Network Peering
- CNI-Genie
- Weave Net Multicast
- kube-bridge

Flannel:

- Flannel is a simple and lightweight network fabric designed for Kubernetes.
- It uses the VXLAN overlay network to create a virtual network connecting nodes.
- Flannel assigns a subnet to each node and ensures that containers on different nodes can communicate with each other.
- It is easy to set up and widely used in Kubernetes clusters.
- Suitable for small to medium-sized clusters and provides good performance.
- GitHub repository: coreos/flannel

Calico:

- Calico provides network policy enforcement and secure network connectivity for Kubernetes.
- It uses standard Linux networking components such as BGP routing and iptables to

enforce policies and route traffic.
- Calico can scale to large clusters and supports advanced networking features.
- It offers network segmentation and isolation, allowing fine-grained control over network traffic between pods and nodes.
- Suitable for large-scale deployments and environments that require strong network policies.
- GitHub repository: projectcalico/calico

Weave:
- Weave provides a simple and flexible network overlay for Kubernetes.
- It creates a virtual network that connects pods across different hosts using VXLAN or UDP tunnels.
- Weave enables direct communication between pods without requiring external load balancers.
- It includes features like network encryption, DNS-based service discovery, and network segmentation.
- Suitable for small to medium-sized clusters and environments that require easy setup and flexible networking options.
- GitHub repository: weaveworks/weave

Cilium:
- Cilium is a networking and security plugin that provides API-aware network and security enforcement for Kubernetes.
- It uses eBPF (extended Berkeley Packet Filter) technology to enable fast packet processing and fine-grained control.
- Cilium offers observability, load balancing, network security policies, and transparent encryption.
- It is suitable for large-scale deployments and environments that require advanced security and observability features.
- GitHub repository: cilium/cilium

Antrea:
- Antrea is a Kubernetes networking plugin specifically designed for Kubernetes native networking.
- It uses Open vSwitch (OVS) as the data plane and supports Kubernetes Network Policy enforcement.
- Antrea leverages the OVS hardware acceleration capabilities for improved performance.
- It provides basic networking and security features while focusing on simplicity and ease of use.
- Suitable for small to medium-sized clusters and environments that prioritize simplicity and Kubernetes-native features.
- GitHub repository: vmware-tanzu/antrea

10. Pod config file

Wednesday, May 24, 2023 9:00 PM

To create a pod (recommended for dev/test environments)
kubectl run --generator=run-pod/v1 nginx-pod --image=nginx

Note: do not run this command directly on production.

The suggested way to create & run a pod in production environment is via YAML config file.

Dummy code:

```
cat nginx-pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
    tier: dev
spec:
  containers:
  - name: nginx-container
    image: nginx
    env:
    - name: DEMO_GREETING
      value: "Hello from the environment"
    ports:
    - containerPort: 80
```

##

Details about the YAML file:
There are various sections of a YAML config file:

1. **apiVersion**
   - *It's a required field.*

- Specifies K8s versioning used.
- Ex:- apiVersion: v1

2. **Kind**
   - *It's a required field.*
   - It defines the kind of object you want to create.
   - Object like: Pod, Deployment, Services.
   - Ex:-
     - Kind: Pod
     - Kind: Deployment
3. **Metadata**
   - *It's a required field.*
   - Contains metadata about the resources like name, label, annotations.
   - This information manages & identifies resources within the cluster.
4. **Spec**
   - *It's a required field*.
   - Defines the desired state & configuration of the resources.
   - Here you define containers, volume, images, ports, environment variables etc.
5. **Container**
   - This comes under the "spec".
   - Here we specify container name, image, env variables, ports & volumes.
6. **Volume**
   - To define persistent storage within resources.
   - This allows you to specifies desired storage volume.
7. **Selector**
   - This comes under "spec".
   - This is used for resources like services, replica set or deployment.

######## these are some of the section of the config file #########

Expanding the YAML services:

Kind	API Version
Pod	v1
ReplicationController	v1
Service	v1
Secret	v1
ServiceAccount	v1
PersistentVolume	v1
PersistentVolumeClaim	v1
ConfigMap	v1
Namespace	v1
ComponentStatus	v1

Kind	API Version
ReplicaSet	apps/v1
Deployment	apps/v1
DaemonSet	apps/v1
Stateful	apps/v1

Kind	API Version
Role	rbac.authorization.k8s.io/v1
RoleBinding	rbac.authorization.k8s.io/v1
ClusterRole	rbac.authorization.k8s.io/v1
ClusterROleBinding	rbac.authorization.k8s.io/v1

Kind	API Version
Job	Batch/v1
Kind	API Version
CronJob	batch/v1beta1

11. Pods - Tasks

Wednesday, May 24, 2023 10:06 PM

POD YAML CODE:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-file-pod
  labels:
    app: nginx
    tier: dev
spec:
  containers:
    - name: nginfile-container
      image: nginx
      env:
        - name: demo_env
          value: "THis is 1st POD config"
      ports:
        - containerPort: 80
```

To create POD using YAML:

```
# kubectl create -f <file>.yaml       //creating object
#kubectl apply -f <file>.yaml         //apply object
```

To display:

```
# kubectl get pods <pod-name>                 //listing all the pods with status
# kubectl get pods <pod-name> -o wide         //listing pods in detailed format
# kubectl get pods <pod-name> -o json         //listing pods info in JSON format
# kubectl get pods <pod-name> -o yaml         //listing pods info in YAML format
```

To describe a pod/useful in troubleshooting

```
# kubectl describe pods <pod-name>
```

To display a pod using label (if added to the YAML config file)

```
# kubectl get pods --show-labels
# kubectl get pods -l app=nginx                 // Print Pods with particular label
```

To edit a pod in a running state:

```
# kubectl edit pods <pod-name>
```

To check logs:

```
# kubectl logs pods <pod-name>
```

To delete a pod:

```
# kubectl delete pods <pod-name>
```

12. Displaying Resource Usage

Wednesday, May 24, 2023 10:16 PM

Installing Metrics-Server:
---
1. Download the metrics-server YAML file:
wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

2. Edit the metrics-server YAML file to add the --kubelet-insecure-tls flag to the metrics-server container command
vi components.yaml

3. Locate the args section under the metrics-server container and add the --kubelet-insecure-tls flag as shown below: (to the bottom of args - line number: 133)
args:
  - --kubelet-insecure-tls

:wq!

4. Apply the metrics-server YAML
kubectl apply -f components.yaml

5. Verify the installation:
kubectl get pods -n kube-system

Give it a minute to gather the data and then run below Top Commands:

Top Command to find the CPU and Memory Usage of Pods, Nodes and Containers:
---
kubectl top pods
kubectl top pods -A
kubectl top pods -A --sort-by memory
kubectl top pods -A --sort-by cpu
kubectl top pods -n [name-space] --sort-by cpu
kubectl top pods -n [name-space] --sort-by memory
kubectl top pods -n [name-space] --sort-by memory > mem-usage.txt

Deleting Pod
~~~~~~~~~~~~~~~~~~
kubectl delete pods <POD-NAME>
~~~~~~~~~~~~~~~~~~

13. ReplicaSet

Wednesday, May 24, 2023 10:41 PM

- A replicaset is a resource object that is used to ensure a specific no. of pod replicas are running & maintained within a cluster.
- It is one of the core controllers provided by K8s & is responsible for managing & scaling pods.
- ReplicaSet always ensures that a specified no. of pods replica are always available & running.
- If a pod fails or terminated, RS automatically replaces it with a new pod to maintain the desired count.
- Similarly, if a count is increased, RS creates additional pods to maintain the desired count.

Key features:

1. Specifying POD template
    - A RS defines a template that specifies the desired config for the pods it manages.
    - The template include details like container, image, command, arguments, environment variables, etc.
2. Specifying replica count
    - A RS "spec.replica" field specifies the desired no. of pods replicas to be maintained by RS.
3. Selector
    - RS uses label selectors to identify the set of pods to manage.
4. Scalability
    - RS support scaling UP or DOWN the no. of replicas by simply modification the "spec.replica" field.

- If replicaset is not mentioned in the config file then default of "1" will be taken & it process the config.

2 major points about replicaset:

1. Earlier we had "Replication Controller" (old version) which was replaced by "ReplicaSet" (new).
2. We don't create replicas & pod manually.
    - This deployment itself creates replicaset in the backend for us.
    - RS then manages the no. of RS.

ReplicaSet vs Replication Controller:

1. apiVersion tag is missing in ReplicaSet (app/v1), not in Replication Controller (v1).
2. A new "matchlabels" in present in ReplicaSet.

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: my-image:latest
        ports:
        - containerPort: 8080
```

Applying the config
kubectl apply -f replicaset.yaml

Listing replicas
kubectl get replicasets my-replicaset

Display ReplicaSet (rs)
kubectl get rs
kubectl get rs <RS-NAME> -o wide
kubectl get rs <RS-NAME> -o yaml
kubectl get rs -l <LABEL>

To increase/decrease number of replicas, need to edit the config file.
vim replicaset.yaml
// change the replicaset number accordingly.
:wq!

OR by using below command (replicaset can be used as rs in short):
kubectl scale rs my-replicaset --replicas=3
kubectl get replicasets my-replicaset

Then apply the changes
kubectl apply -f replicaset.yaml

& verify
# kubectl get replicasets my-replicaset

Print Details of ReplicaSet
kubectl describe rs <RS-NAME>

To check if replicaset is working or not
kubectl get pods

Scaling Applications
kubectl scale rs <RS-NAME> --replicas=[COUNT]

Editing ReplicaSet
kubectl edit rs <RS-NAME>

Then delete one pod from this list
kubectl delete pod my-replicaset-pw449

Check replicaset again
kubectl get replicasets my-replicaset //here status & desired will be as old config.

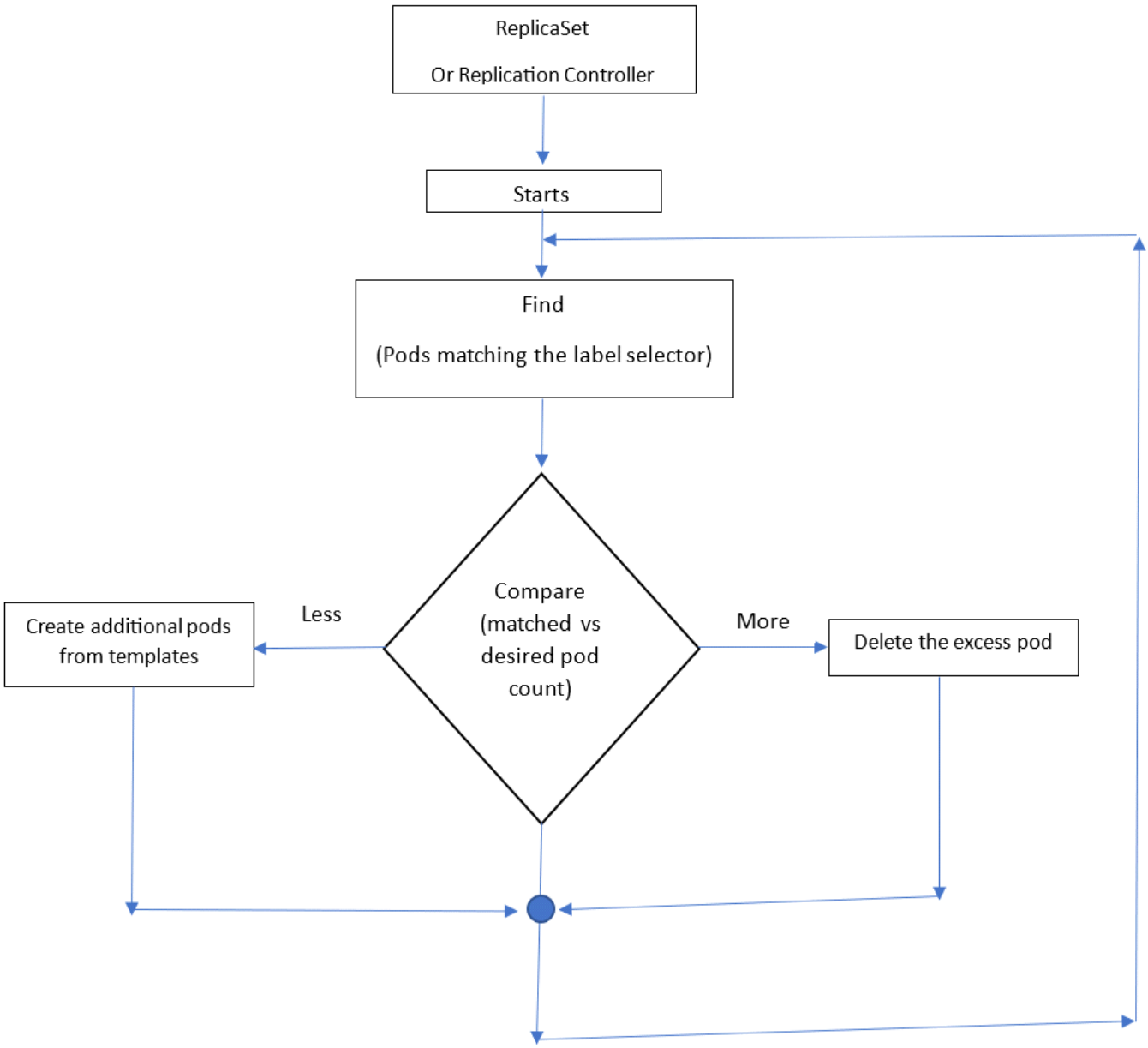
Check the pod status
kubectl get pods

Deleting ReplicaSet
kubectl delete rs <RS-NAME>

14. Replication loop

Thursday, May 25, 2023 6:19 PM

- Replication controller & replicaset are based on replication loop.
- It's basically a loop.



15. ReplicaSet YAML file

Thursday, May 25, 2023 6:38 PM

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
        matchExpressions:
          - {key:tier, operator:In, values:[frontend]}
  template:
    metadata:
         name: my-pod
      labels:
        app: my-app
        tier: frontend
    spec:
      containers:
      - name: my-container
        image: nginx:latest
        ports:
        - containerPort: 80
```

```
Simple replica set example:
---------------------------
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

apiVersion	API version of resource type
Kind:ReplicaSet	Resource type
Replicas	3
{key:tier, operator:In, values:[frontend]}	Here we are using set-based operator

template:
  metadata:
  name: my-pod
  labels:
  app: my-app
  tier: frontend
  spec:
  containers:
  - name: my-container
  image: nginx:latest
  ports:
  - containerPort: 80
	POD template used for creating new pods.

To list labels:
kubectl get rs --show-labels

16. ReplicaSet Tasks

Thursday, May 25, 2023 6:51 PM

ReplicaSet YAML code:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

Creating/Applying replica config file

```
# kubectl create -f [file-name].yaml
# kubectl apply -f [file-name].yaml
```

Displaying the details of replica set (replica set == rs)

```
# kubectl get rs <replication-name>
# kubectl get rs <replication-name> -o wide
# kubectl get rs <replication-name> -o json
# kubectl get rs <replication-name> -l [KEY=NAME]
```

Describing a replica set

```
# kubectl describe rs <replication-name>
```

Scaling a replica set

```
# kubectl scale rs <replication-name>--replicas=<COUNT>
```

Editing a replica set

```
# kubectl edit rs <replication-name>
```

Deleting a replica set

```
# kubectl delete rs <replication-name>
```

17. Namespaces

Thursday, May 25, 2023 6:57 PM

- It's a way of partitioning the entire K8s cluster into multiple virtual partitions.
- You can partition your K8s cluster into:
  - Various teams
  - Applications
  - Environments
  - Or by any other custom requirement.
- By default, k8s install 2 namespaces
  1. Kube-system namespace
     - This contains all the system related pods like: API-server, controller master, ETCD, schedular, Kube-proxy.
  2. Default namespace
     - We create all the K8s objects under this & manage it.

To create a custom namespace

```
# kubectl create namespace dev
# kubectl create namespace prod
```

- In these namespaces, we can create 2 resources with identical configurations & identical names, if required & it one does not affect other.

Note:- Nodes are shared across cluster. They are not restricted to a single namespace.

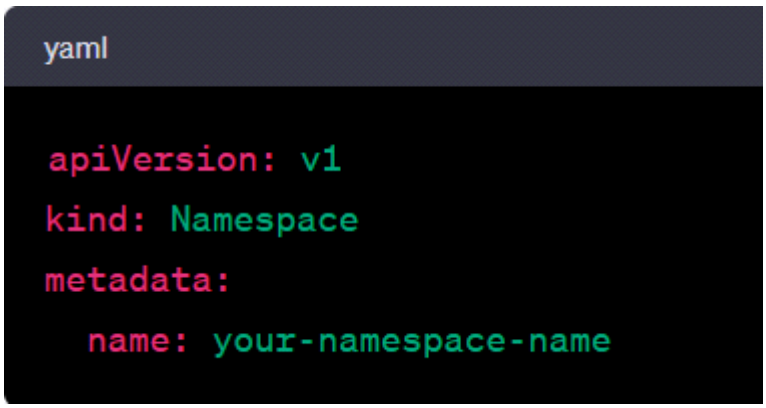
Namespaces are highly effective between apps, teams & environments.

18. Namespace YAML config

Friday, May 26, 2023 2:29 PM

YAML configuration file for creating a namespace in Kubernetes:

```
apiVersion: v1
kind: Namespace
metadata:
  name: your-namespace-name
```



To create a namespace using this configuration file

```
# kubectl apply -f namespace.yaml
```

Instead we can also use:

```
# kubectl create namespace dev
# kubectl create namespace qa
# kubectl create namespace prod
```

To create a reason in a specific namespace:

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    env: dev
  name: nginx-pod
  namespace: dev
spec:
  …
```

Namespace - needs to be added/applied on metadata of the config file.

To list the resources/pods running in a specific NS.

```
# kubectl get pods --namespace=dev
```

19. Namespace - Tasks

Friday, May 26, 2023 2:38 PM

Listing all namespaces:

```
# kubectl get namespaces
```

Creating namespace using file:

```
# kubectl create -f filename.yaml
# kubectl apply -f filename.yaml
```

Creating NS using cmd:

```
# kubectl create namespace <ns-name>
```

Display namespace (ns):

```
# kubectl get ns <ns-name>
# kubectl get ns -o wide
# kubectl get ns -o yaml
# kubectl get pods --namespace=<name>
```

Describe:

```
# kubectl describe ns <name>
```

Set as default namespace:

```
# kubectl config set-context -current --namespace=<name>
# kubectl config view | grep namespace
```

Updating the namespace config:

```
# kubectl edit ns <name>
```

Deleting namespace:

```
# kubectl delete ns <name>
```

Command to create a pod in custom namespace:

```
# kubectl run nginx --image=nginx --namespace=dev
```

Validate:

```
# kubectl get pods
# kubectl get pods -n dev
```

20. Upgrading K8s version using Kubeadm

Saturday, May 27, 2023 10:07 PM

Installation can be done in 2 ways:

1. For **Managed service**:- GKE, AKS, EKS will take care of updates & upgrades.
2. For **Self-managed service**:- administrator needs to take care of all updates & upgrades.

- Minor version should be upgraded in sequence.
- For ex:-
  - Current version: 1.16.0
  - Latest version: 1.18.0
    - So we need to upgrade to 1.17.0 then 1.18.0

- Standard approach is to upgrade.
  - Control-Plane (first) (one-node at a time).
  - Then worker node.

- On Master & Worker node upgrade approach is
  a) Kubeadm upgrade
  b) Kubernetes upgrade
  c) Kubelet & Kubectl upgrade

##
READ BEFORE YOU RUN:
~~~~~~~~~~~~~~~~~~~~

a. These commands need to run on cluster running with Ubuntu
b. Master and Worker nodes in your cluster might differ from this.
c. Kubernets version might be different in your case, but procedure is the same.
d. For more details, please visit Kubernetes docs.

****
****

OVERVIEW: Steps involved in Kubernetes Version Upgrade:
~~~~~~

1. Upgrade kubeadm
2. Upgrade Node
3. Drain the node (before kubelet upgrade)
4. Upgrade kubelet and kubectl
5. Restart the kubelet
6. Uncordon the node
---
7. Validate the Upgrade

****

A. Upgrading MASTER(Control-Plane) Node(1).

_____

0. Determine which version to upgrade to
---
apt update
apt-cache madison kubeadm

1. Ugrading Kubeadm:

NOTE: If a package is marked "hold", it is held back: The package cannot be installed, upgraded, or removed until the hold mark is removed.

apt-mark unhold kubeadm && \
apt-get update && apt-get install -y kubeadm=1.21.0-00 && \
apt-mark hold kubeadm

kubeadm version

~~~~~~

2. Upgrading Node:
---
kubeadm upgrade plan

kubeadm upgrade apply v1.21.0
~~~~~~

~~~~~~

3. Drain the node:

Prepare the node for maintenance by marking it unschedulable and evicting the worklo

kubectl drain master --ignore-daemonsets

~~~~~~

4. Upgrade "kubelet" and "kubectl":
---

apt-mark unhold kubelet kubectl && \
apt-get update && apt-get install -y kubelet=1.21.0-00 kubectl=1.21.0-00 && \
apt-mark hold kubelet kubectl

~~~~~~

5. Restart the "Kubelet":
---
systemctl daemon-reload
systemctl restart kubelet

~~~~~~

6. Uncordon the node:
---
kubectl uncordon master

****
B). Upgrading WORKER Node(1):
_____

The upgrade procedure on worker nodes should be executed one node at a time or few nodes at a time, without compromising the minimum required

capacity for running your workloads.

1. Upgrade kubeadm:
---
apt-mark unhold kubeadm && \
apt-get update && apt-get install -y kubeadm=1.21.0-00 && \
apt-mark hold kubeadm

~~~~~~

2. Upgrade "Node": (Run it on worker node)

NOTE: For worker nodes this upgrades the local kubelet configuration:

kubeadm upgrade node

~~~~~~

3. "Drain" the node: (Run it on master node)

Prepare the node for maintenance by marking it unschedulable and evicting the workloads:

kubectl drain worker-1 --ignore-daemonsets

~~~~~~

4. Upgrade kubelet and kubectl:
---

apt-mark unhold kubelet kubectl && \
apt-get update && apt-get install -y kubelet=1.21.0-00 kubectl=1.21.0-00 && \
apt-mark hold kubelet kubectl

~~~~~~

5. Restart the kubelet:
---

systemctl daemon-reload
systemctl restart kubelet

6. Uncordon the node:

Bring the node back online by marking it schedulable:

kubectl uncordon worker-1

****

C). Upgrading WORKER Node(2):
_____

Follow same above 6 steps as mentioned in "Upgrading WORKER Node(1)

****
D. Verify the status of the cluster
_____

After the kubelet is upgraded on all nodes verify that all nodes are available again by running the following command from anywhere kubectl can access the cluster:

kubectl get nodes
kubeadm version
kubectl version

****

##########

Connect to Kubernetes 1st Master node:

A) Upgrading Kubeadm using commands

yum install -y kubeadm-1.17.0 --disableexcludes=kubernetes

- For ubuntu

apt-get update -y

apt-get upgrade kubeadm -y //need to test this on ubuntu.

- Validating

kubeadm version

- Drain / Make node un-schedulable & gracefully terminate pods

kubectl drain [node-name] --ignore -daemonsets

B) Upgrading kubernetes

- For this, we need to take 2 steps:
  1. We need to PLAN.
  2. We need to APPLY.

Planning

kubeadm upgrade plan

Apply

kubectl upgrade apply v1.17.0

C) Upgrading kubectl & kubelet & then restart the service.

- On CentOS

yum install -y kubelet-1.17.0 kubectl-1.17.0 --disableexcludes=kubernetes

systemctl restart kubelet

- On ubuntu

NEED TO SEARCH

- Making node schedulable again:

kubectl uncordon <master-1>

Connect to Kubernetes 2nd Master node:

Plan

```
#  kubeadm upgrade plan
```

Upgrading kubectl & kubelet & then restart the service.

- On CentOS

```
# yum install -y kubelet-1.17.0 kubectl-1.17.0 --
disableexcludes=kubernetes
# systemctl restart kubelet
```

############## THAT'S IT ##################

Connect to Kubernetes 1st Worker node:

---

1. Upgrade kubeadm
   # yum install -y kubeadm-1.17.0 --disableexcludes=kubernetes

2. Validate kubeadm version
   # kubeadm version

3. Drain & make worker unusable (un-schedulable)
   # kubectl drain [node] --ignore-daemonsets

4. Now upgrade node
   # kubeadm upgrade node

5. Upgrade kubectl & kubelet & restart service
   # yum install -y kubelet-1.17.0 kubectl-1.17.0 --
   disableexcludes=kubernetes
   # systemctl restart kubelet

6. Make the node usable / schedulable
   # kubectl uncordon worker-1

Note: Exact same steps (from step 1 - step 6) are required in all the worker nodes.

21. ETCD backup & restore - Tested

Saturday, May 27, 2023 11:03 PM

- ETCD is key-value database.
- It stores:
  - Current set of pods.
  - Deployment
  - Replicasets
  - ConfigMaps
  - Jobs

  & all other configs as key-value pairs.

Backup approach:

1. Managed service
   a. GKE, AKS, EKS will take care of all backup & updates.
2. Self-managed service
   a. You have to take the backup on regular basis (manually).

Why need to take backup?

1. When there is a "disaster".
   - Ex:- when someone accidently delete the namespace, where all your data resides

2. When you want to replicate the entire environment.
   - Ex:- you want to replicate entire production env. to staging env.

3. When you want to migrate.
   - Ex:- when you want to migrate one environment to another.

We will be using "etcdctl" tool, which allows us to

- Add new entries
- Delete entries
- Take snapshots of entire K8s cluster.
- Restore etcd db.
- Using commands like:
  - Put, get, save, restore, status.

If it's not available, then download it from
https://github.com/etcd-io/etcd/releases (link working, tested)

. PRE-REQ: Installing "etcdctl" client:

Ensure if "etcdctl" command line tool is available by running below command.

export ETCDCTL_API=3
etcdctl version

If not available, then you can do the same by following below steps.

---

mv /tmp/etcd-download-test/etcdctl /usr/bin

ETCD_VER=v3.4.14

# choose either URL
GOOGLE_URL=https://storage.googleapis.com/etcd
GITHUB_URL=https://github.com/etcd-io/etcd/releases/download
DOWNLOAD_URL=${GOOGLE_URL}

rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
rm -rf /tmp/etcd-download-test && mkdir -p /tmp/etcd-download-test

curl -L ${DOWNLOAD_URL}/${ETCD_VER}/etcd-${ETCD_VER}-linux-amd64.tar.gz -o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp/etcd-download-test --strip-components=1
rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz

mv /tmp/etcd-download-test/etcdctl /usr/bin

Reference: https://github.com/etcd-io/etcd/releases

****

1. Deploy sample Pod for testing:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl run nginx-pod --image=nginx

****

##
Create a pod:
################
kubectl run nginx-pod --image=nginx
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

validate :
################
kubectl get pods

To take snapshot:
################

ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
 --cacert=<trusted-ca-file> --cert=<cert-file> --key=<key-file> \
 snapshot save <backup-file-location>

to find the location of CA file
kubectl -n kube-system describe pod etcd-kmaster
    -> --cacert=/etc/kubernetes/pki/etcd/ca.crt

to find cert file
-> --cert=/etc/kubernetes/pki/etcd/server.crt

to find the key file
-> --key=/etc/kubernetes/pki/etcd/server.key

backup location
-> /opt/snapshot-pre-boot.db

ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \
 snapshot save /opt/snapshot-pre-boot.db

delete the pod:
################
kubectl delete pod nginx-pod

Now to restore:
################
ETCDCTL_API=3 etcdctl --data-dir=/var/lib/etcd2 snapshot restore /opt/snapshot-pre-boot.db

Now edit the etcd config file:
################################
vim /etc/kubernetes/manifests/etcd.yaml
    on line 79:
    path: /var/lib/etcd2

:wq!

Wait for some time so that the config can link & check the get you deleted in the beginning.:
################
kubectl get pods

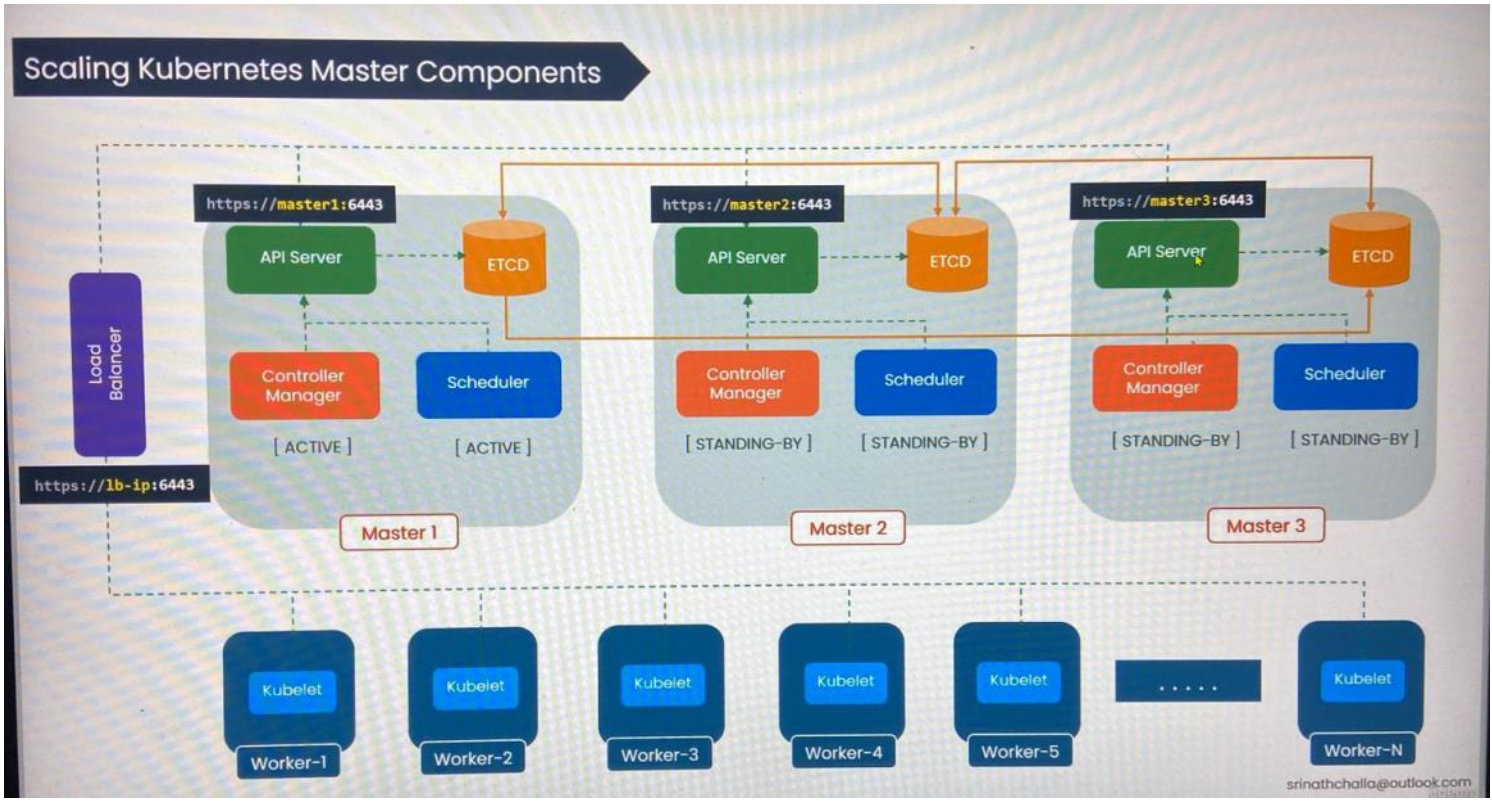
##

kubectl get pod nginx-pod

****

22. Managing a highly-available K8s cluster

Saturday, May 27, 2023 11:26 PM



Generally, it is recommended to have master server in odd numbers (like 3, 5, 7, 9, 11) in self-managed K8s environment.

Here,

ETCD

- Must be aware of one another for read/write.
- It has the ability to run multiple instances.
- All you need to do is run multiple instances of machines 3,5,7,9,11.
- We have to specify in config file for ETCD.

API-SERVER

- It is stateless (much easier).
- You can run as many APISERVERS as you need.
- Each API-SVR is co-located with etcd.by doing this, ETCD does not need a load balancer (LB) in front of them because all API-SERVERS only talk to its local ETC instances.
- API-SERVER do need a LB so that clients (like kubectl, controller manager & all kubelets) connects to only the healthy API-servers.

Controller manager & Schedular

- Very difficult to make them HA because controllers & schedulers all watches cluster states & acts when it changes which modifies cluster state further.
- Ex:- if a replica set is updated

23. Role & Role-binding - tested

Wednesday, May 31, 2023 9:33 PM

0. **Create a user "appuser".**
   # useradd appuser

1. Creating Kubernetes test User Account(appuser) (using x509 for testing RBAC)

 # Generating Key
openssl genrsa -out appuser.key 2048

 # Generaing Certificate Signing request (csr):
openssl req -new -key appuser.key -out appuser.csr -subj "/CN=appuser"

 # Singing CSR using K8s Cluster "Certificate" and "Key"
openssl x509 -req -in appuser.csr \
-CA /etc/kubernetes/pki/ca.crt \
-CAkey /etc/kubernetes/pki/ca.key \
-CAcreateserial \
-out appuser.crt -days 300

 # Adding user credentials to "kubeconfig" file
kubectl config set-credentials appuser --client-certificate=appuser.crt --client-key=appuser.key

 # Creating context for this user and associating it with our cluster:
kubectl config set-context appuser-context --cluster=kubernetes --user=appuser

 # Displaying K8s Cluster Config
kubectl config view

****
****

2. Creating Namespaces and Pod for testing RBAC:

2a. Creating test Namespace:
---
kubectl create ns dev-ns

.....

1b. Creating test Pod:

---

kubectl run nginx-pod --image=nginx -n dev-ns
kubectl get pods -n dev-ns

.....

1c. Test Before Deploying (gives error):

---

kubectl get pods -n dev-ns --user=appuser

****

2. Creating a "Role" & "RoleBinding":

2a. Creating Resources Declaratively (Using YAML):

```
# Role
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: dev-ns
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
---
# RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: dev-ns
subjects:
- kind: User
  name: appuser
  apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

.....
Verify the role exists or not:
kubectl get role -n dev-ns

Apply the config
kubectl apply -f rolebind.yaml

Verify the role again:
kubectl get role -n dev-ns

2b. Creating Resources Imperatively (Commands):

```
# role
kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods --namespace=dev-ns

# rolebinding
kubectl create rolebinding read-pods --role=pod-reader --user=appuser --namespace=dev-ns
```

run below command, that gave error earlier:
kubectl get pods -n dev-ns --user=appuser

****

3. Display Role and RoleBinding:

```
# role
kubectl get role -n dev-ns

# rolebinding
kubectl get rolebinding -n dev-ns

kubectl describe role -n dev-ns
kubectl describe rolebinding -n dev-ns
```

****

4. Testing RBAC:

Pod Operations: get, list, watch - in "dev-ns" namespace:

---

kubectl auth can-i get pods -n dev-ns --user=appuser // Yes
kubectl auth can-i list pods -n dev-ns --user=appuser // Yes
kubectl auth can-i delete pods -n dev-ns --user=appuser // No
kubectl auth can-i update pods -n dev-ns --user=appuser // No

kubectl get pod nginx-pod -n dev-ns --user=appuser
kubectl get pods -n dev-ns --user=appuser

.....

Pod Operations: get, list, watch - in "NON dev-ns" namespace:

---

kubectl auth can-i get pods -n kube-system --user=appuser
kubectl auth can-i list pods -n kube-system --user=appuser
kubectl auth can-i watch pods -n kube-system --user=appuser

Shows error for below cmds:

kubectl get pods --user=appuser # queries default namespace
kubectl get pods -n kube-system --user=appuser

.....

Creating Objects in "dev-ns" namespace:

---

kubectl auth can-i create pods -n dev-ns --user=appuser
kubectl auth can-i create services -n dev-ns --user=appuser
kubectl auth can-i create deployments -n dev-ns --user=appuser

kubectl run redis-pod -n dev-ns --image=redis --user=appuser
kubectl create deploy redis-deploy -n dev-ns --image=redis --user=appuser

.....

****

5. Cleanup:

---

kubectl config unset contexts.appuser-context

```
kubectl config unset users.appuser

kubectl config view

kubectl get pod nginx-pod -n dev-ns --user=appuser
kubectl get pods -n dev-ns --user=appuser

kubectl delete role pod-reader -n dev-ns
kubectl delete rolebinding read-pods -n dev-ns
```

24. ClusterRole & ClusterRoleBinding

Wednesday, May 31, 2023 9:46 PM

1. Creating Kubernetes test User Account(appmonitor) (using x509 for testing RBAC)

 # Generating Key

openssl genrsa -out appmonitor.key 2048

 # Generaing Certificate Signing request (csr):

openssl req -new -key appmonitor.key -out appmonitor.csr -subj "/CN=appmonitor"

 # Singing CSR using K8s Cluster "Certificate" and "Key"

```
openssl x509 -req -in appmonitor.csr \
    -CA /etc/kubernetes/pki/ca.crt \
    -CAkey /etc/kubernetes/pki/ca.key \
    -CAcreateserial \
    -out appmonitor.crt -days 300
```

 # Adding user credentials to "kubeconfig" file

kubectl config set-credentials appmonitor --client-certificate=appmonitor.crt --client-key=appmonitor.key

 # Creating context for this user and associating it with our cluster:

kubectl config set-context appmonitor-context --cluster=kubernetes --user=appmonitor

 # Displaying K8s Cluster Config

kubectl config view

****

2. Creating Namespaces and Pod for testing RBAC:

---

kubectl create ns test-ns1
kubectl create ns test-ns2

2a. Creating test Pod:

#Pod
kubectl run nginx-pod-default --image=nginx
kubectl run redis-pod-ns1 --image=redis -n test-ns1
kubectl run httpd-pod-ns2 --image=busybox -n test-ns2

.....

2c. Test Before Deploying (gives error):

---
kubectl get pods --user=appmonitor
kubectl get pods -n test-ns1 --user=appmonitor
kubectl get pods -n test-ns2 --user=appmonitor
kubectl get pods -n kube-system --user=appmonitor
kubectl get pods -A --user=appmonitor

****
****

2. Creating a "ClusterRole" & "ClusterRoleBinding":

2a. Creating Resources Declaratively (Using YAML):

ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: clusterrole-monitoring
rules:

```
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
---
# ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: clusterrole-binding-monitoring
subjects:
- kind: User
  name: appmonitor
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: clusterrole-monitoring
  apiGroup: rbac.authorization.k8s.io
```

apply role

```
# kubectl apply -f clusterrole.yaml
```

Test Before Deploying (No error this time):

```
-------------------------
kubectl get pods --user=appmonitor
kubectl get pods -n test-ns1 --user=appmonitor
kubectl get pods -n test-ns2 --user=appmonitor
kubectl get pods -n kube-system --user=appmonitor
kubectl get pods -A --user=appmonitor
.....................................................................
```

2b. Creating Resources Imperatively (Commands):

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

 # Cluster-role

```
kubectl create clusterrole clusterrole-monitoring --
verb=get,list,watch --resource=pods
```

 # Cluster-rolebinding
kubectl create clusterrolebinding clusterrole-binding-monitoring --clusterrole=clusterrole-monitoring --user=appmonitor

****

3. Display ClusterRole and ClusterRoleBinding:

 # clusterrole
kubectl get clusterrole | grep clusterrole-monitoring

 # clusterrolebinding
kubectl get clusterrolebinding | grep clusterrole-binding-monitoring

kubectl describe clusterrole clusterrole-monitoring
kubectl describe clusterrolebinding clusterrole-binding-monitoring

****

4. Testing ClusterRole & ClusterRoleBinding:

Pod Operations: get, list, watch - in "kube-system", "default", "test-ns1", and "test-ns2" namespaces:
---
kubectl auth can-i get pods -n kube-system --user=appmonitor
kubectl auth can-i get pods -n default --user=appmonitor
kubectl auth can-i get pods -n test-ns1 --user=appmonitor
kubectl auth can-i get pods -n test-ns2 --user=appmonitor

.....

Creating Objects in "default" (or in any other) namespace: -->

Shows NO

---

kubectl auth can-i create pods --user=appmonitor
kubectl auth can-i create services --user=appmonitor
kubectl auth can-i create deployments --user=appmonitor

kubectl run redis-pod --image=redis --user=appmonitor
kubectl create deploy redis-deploy --image=redis --user=appmonitor

.....

Deleting Objects in "default" (or in any other) namespace: -->
Shows NO

---

kubectl auth can-i delete pods --user=appmonitor
kubectl auth can-i delete services --user=appmonitor
kubectl auth can-i delete deployments --user=appmonitor

kubectl delete pods nginx-pod --user=appmonitor

****
****

5. Cleanup:

#Delete ClusterRole and ClusterRoleBinding:

---

kubectl delete clusterrole clusterrole-monitoring
kubectl delete clusterrolebinding clusterrole-binding-monitoring

#Removing User and Context from Cluster Config

---

kubectl config unset users.appmonitor
kubectl config unset contexts.appmonitor-context

Ensure user "appmonitor" and its configuration is removed:

---
kubectl get pods --user=appmonitor
kubectl config view

Deleting Pods:

---
kubectl delete pod nginx-pod-default
kubectl delete pod redis-pod-ns1 -n test-ns1
kubectl delete pod httpd-pod-ns2 -n test-ns2

#Deleting Namespace:

---
kubectl delete ns test-ns1
kubectl delete ns test-ns2

#Validating:

---
kubectl get ns
kubectl get pods
kubectl get clusterrole | grep monitoring
kubectl get clusterrolebinding | grep monitoring

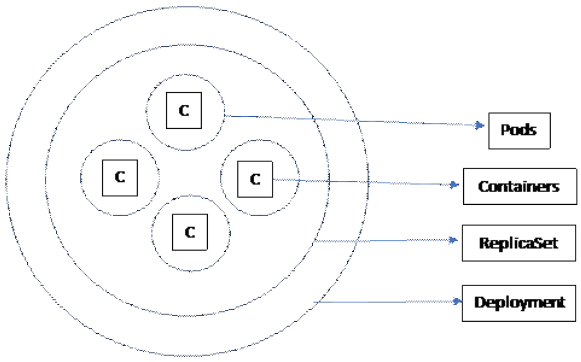
****
****

25. Deployment

Tuesday, June 6, 2023 7:38 PM

Whenever an app is deployed, there are few things to consider.

- Scaling of app.
  - Meaning, increasing & decreasing of instances as per configuration.
- Rollout & Rollback of version.
  - 'Deployment' will take care of rollout & rollback.



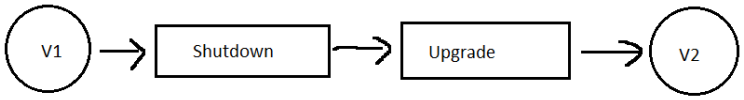
Deployment features:

- Deployment supports following features under 2 categories.

Application deployment	Rollout
	Pause & Resume
	Rollback
Scaling	Replica
	Scale up & scale down

Deployment strategy:

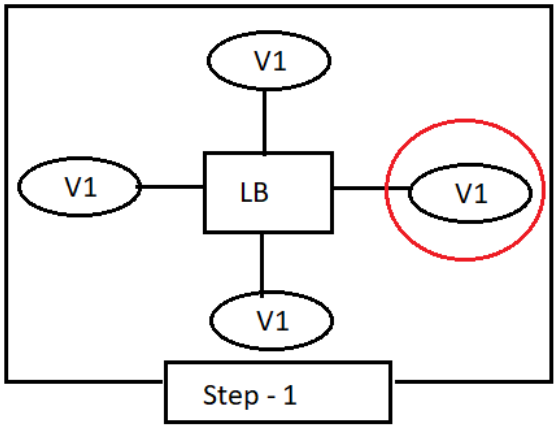
1. *Recreate*
   - Aka dummy deployment
   - We will shut down all deployment of "V1" & 2 more "V2" while upgrading V1 to V2.
   - Here, at a time 2 pods will not be available (out of 4).
   - Simple to use.
   - Downtime - while migrating v1 to v2.
   - This is ideal for DEV, QA environments.

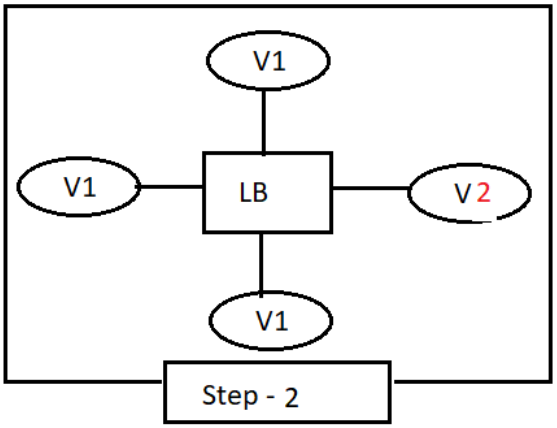


2. *Rolling updates*
   - K8s slowly rollout updates to all the instances one by one.
   - It's a time consuming process.
   - This is a default strategy in K8s.
   - Working:
     - When a pod goes under upgradation (Out of 4 pods), one pod with V1 is terminated & traffic get stopped towards this pod.
     - Then a new pod with V2 will be spin & take its place.
     - Meanwhile remaining V1 pods receives updates.
     - Once this V2 pod is ready, the same procedure is applied to other 3 pods.
     - When V2 is ready K8s will start sending the traffic to this

V2.

- This will continue until all pods are replaced by V2 pods.





3. *Canary deployment*
   - We use this to test new version before it is rolled out.
   - Users can rollout if there are issues with the new version.
   - It is a slow rollout process.
4. *Blue/Green upgrade strategy*
   - We deploy same no. of never version instances along with the older version.
   - Here goal is to switch the traffic from older version to newer version.
   - In case if something is wrong with newer version, it is easy to roll back older version, else we remove older version.
   - It is very expensive as it requires double resources.

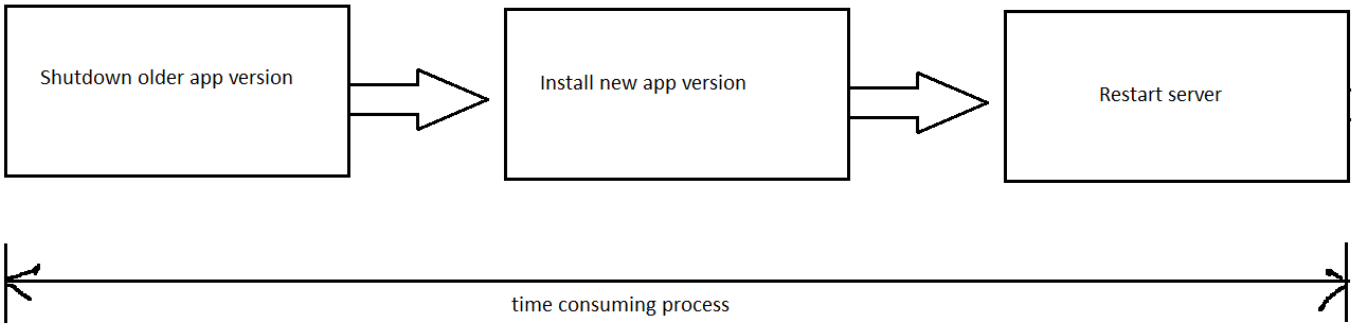
Command to check the deployment strategy using kubectl command:

```
# kubectl describe deploy nginx | grep StrategyType
```

26. Deployment - Strategy

Tuesday, June 6, 2023 8:48 PM

Older method to upgrade:



Newer method to upgrade:

- Rollout updates to all instances one by one.
- No interruption while updating.
- This is called "**zero-down time**" deployment.

27. Deployment - YAML

Wednesday, June 7, 2023 9:37 AM

Creating Deployment Declaratively (Using YAML file)

```
# nginx-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.18
        ports:
        - containerPort: 80
```

Deployment of YAML:

```
# kubectl apply -f nginx-deploy.yaml
```

Creating Deployment "Imperatively" (from command line):

```
kubectl create deployment NAME --image=[IMAGE-NAME] --replicas=[NUMBER]
```

Displaying Deployment

kubectl get deploy <NAME>
kubectl get deploy <NAME> -o wide
kubectl get deploy <NAME> -o yaml

Describing Deployment
~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl describe deploy <NAME>

Print Details of Pod Created by this Deployment
~~~~~~

kubectl get pods --show-labels
kubectl get pods -l [LABEL]

EX: kubectl get pods -l app=nginx-app

Scaling Applications:
~~~~~~~~~~~~~~~~~~~~~~~~

kubectl scale deploy [DEPLOYMENT-NAME] --replicas=[COUNT]
Update the replica-count to 5

Edit the Deployment:
~~~~~~~~~~~~~~~~~~~~~~~

kubectl edit deploy [DEPLOYMENT-NAME]

Running operations directly on the YAML file:
~~~~~~

SYNTAX: kubectl [OPERATION] –f [FILE-NAME.yaml]

kubectl get –f [FILE-NAME.yaml]
kubectl describe –f [FILE-NAME.yaml]
kubectl edit –f [FILE-NAME.yaml]
kubectl delete –f [FILE-NAME.yaml]
kubectl create –f [FILE-NAME.yaml]

Delete the Deployment:
~~~~~~~~~~~~~~~~~~~~~~~~

kubectl delete deploy <NAME>

```
kubectl get deploy
kubectl get rs
kubectl get pods
```

28. Rollout & Rollback upgrades

Wednesday, June 7, 2023 11:00 AM

In this Demo:

We will create deploy sample application. Next we will update deployment by setting new Image version. What if we incorrectly put Image version? we will roll out. You will also learn about rolling back.

****

Creating Deployment "Imperatively" (from command line):

~~~~~~

kubectl create deployment NAME --image=[IMAGE-NAME] --replicas=[NUMBER]

EX:

kubectl create deployment nginx-deploy --image=nginx:1.18 --replicas=4

****

Upgrading Deployment with new Image:

~~~~~~

kubectl set image deploy [DEPLOYMENT-NAME] [CONTAINER-NAME]=[CONTAINER-IMAGE]:[TAG] --record

EX:

kubectl set image deploy nginx-deploy nginx=nginx:1.91 --record

Checking Rollout Status:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl rollout status deploy [DEPLOYMENT-NAME]

EX:

kubectl rollout status deploy nginx-deploy

Waiting for deployment "nginx-deploy" rollout to finish: 2 out of 4 new replicas have been updated…

NOTE: There is some issue. To dig deep, let's check rollout history.

Checking Rollout History:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl rollout history deploy [DEPLOYMENT-NAME]

EX:

root@master:~# kubectl rollout history deploy nginx-deploy

deployment.apps/nginx-deploy

REVISION CHANGE-CAUSE

1 kubectl set image deploy nginx-deploy nginx-container=nginx:1.91 --record=true

2 kubectl set image deploy nginx-deploy nginx=nginx:1.91 --record=true

NOTE: From the output, you can see the commands that are run previously.
If you notice, Image tag we used is 1.91 instead of 1.19. Let's rollback!

You can confirm the same from by running

kubectl get deploy nginx-deploy -o wide

Doing previous rollout "undo":
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl rollout undo deployment/[DEPLOYMENT-NAME]
(OR)
kubectl rollout undo deployment [DEPLOYMENT-NAME] --to-revision=[DESIRED-REVISION-NUMBER]

kubectl rollout status deployment/[DEPLOYMENT-NAME]

kubectl get deploy [DEPLOYMENT-NAME] -o wide

Doing Rollout with correct Image version:
~~~~~~

kubectl set image deploy [DEPLOYMENT-NAME] [CONTAINER-NAME]=[CONTAINER-IMAGE]:[TAG] --record

kubectl rollout status deploy [DEPLOYMENT-NAME]

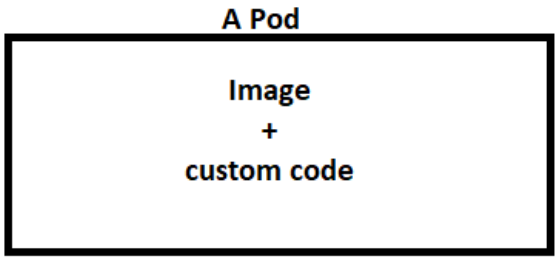
kubectl get deploy [DEPLOYMENT-NAME] -o wide

29. ConfigMaps

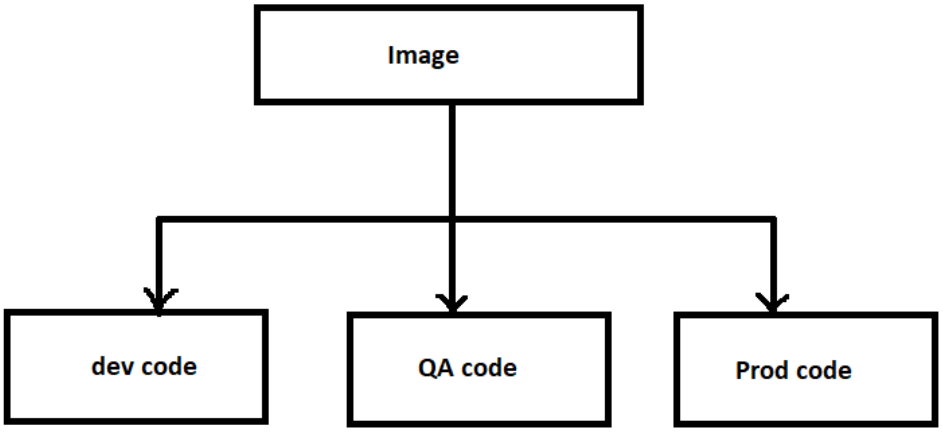
Wednesday, June 7, 2023 10:46 AM

- It is an API-object to store custom configuration data outside of your pod-object.
- This data should be "non-sensitive data".
- Sensitive data like certs, usernames, passwords shouldn't be part of ConfigMaps because this data on ConfigMaps will be shared with other teams.
- Assume you want to run containers in different environments with the same image but with different configurations inside the image.

Generally,



For different environments:



ConfigMaps can allow you to decouple env-specific configurations from your contain image. So that your apps are easily portable.

ConfigMaps can mount data to the pods & containers using:

1. Key-value pair.
2. File with the data in it.

ConfigMaps are used as key-value pairs.

Key1:Value1
Key2:Value2

ConfigMaps as file

Key: nginx.conf
value:
http{
…
}

These ConfigMaps could be the used as

1. Environment variables
2. Arguments
3. Files in volumes

30. ConfigMaps - Tasks

Wednesday, June 7, 2023 10:49 AM

Creating ConfigMaps Declaratively (Using YAML file):

Example-1:

```
---------
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-config-yaml
data:
  ENV_ONE: "va1ue1"
  ENV_TWO: "va1ue2"
==============================
[root@kmaster1 ~]$ kubectl apply -f configMaps.yaml
configmap/env-config-yaml created

[root@kmaster1 ~]$ kubectl get configmaps
NAME              DATA   AGE
env-config-yaml   2      2m21s
kube-root-ca.crt  1      5d14h

[root@kmaster1 ~]$ kubectl describe configmaps env-config-yaml
Name:         env-config-yaml
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
ENV_TWO:
----
va1ue2
ENV_ONE:
----
va1ue1

BinaryData
====

Events:  <none>
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-nginx-config-yaml
data:
  my-nginx-config.conf: |-
    server {
      listen 80;
      server_name www.jeetusingh.in;
      gzip on;
      gzip_types text/plain application/xml;
      location / {
```

```
      root /usr/share/nginx/html;
      index index.html index.htm;
    }
  }
  sleep-interval: 25
```

Creating ConfigMap Imperatively (from Command line):
~~~~~~

kubectl create configmap <NAME> <SOURCE>

From Literal value:
----

kubectl create configmap env-config-cmd --from-literal=ENV_ONE=value1 --from-literal=ENV_TWO=value2

From File:
----

kubectl create configmap my-ngix-config-file-cmd --from-file=/path/to/configmap-file.txt

kubectl create configmap my-config --from-file=path/to/bar

Displaying ConfigMap:
~~~~~~~~~~~~~~~~~~~~~~~~~~~

kubectl get configmap <NAME>
kubectl get configmap <NAME> -o wide
kubectl get configmap <NAME> -o yaml
kubectl get configmap <NAME> -o json
kubectl describe configmap <NAME>

Editing ConfigMap:
~~~~~~~~~~~~~~~~~~~~~~~~

kubectl edit configmap <NAME>

Injecting ConfigMap into Pod As Environment Variables

```
# cm-pod-env.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cm-pod-env
spec:
  containers:
    - name: test-container
      image: nginx
      env:
      - name: ENV_VARIABLE_1
        valueFrom:
          configMapKeyRef:
            name: env-config-yaml
            key: ENV_ONE
      - name: ENV_VARIABLE_2
```
~~~~~~~~~~~~~~~~~~~~~~~~

```
      valueFrom:
        configMapKeyRef:
          name: env-config-yaml
          key: ENV_TWO
  restartPolicy: Never


Deploy:
-------
kubectl apply -f cm-pod-env.yaml

Validate:
---------
kubectl exec cm-pod-env -- env | grep ENV
```

Injecting ConfigMap into Pod As Arguments(2/2)

```
# cm-pod-arg.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cm-pod-arg
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "echo $(ENV_VARIABLE_1) and $(ENV_VARIABLE_2)" ]
      env:
      - name: ENV_VARIABLE_1
        valueFrom:
          configMapKeyRef:
            name: env-config-yaml
            key: ENV_ONE
      - name: ENV_VARIABLE_2
        valueFrom:
          configMapKeyRef:
            name: env-config-yaml
            key: ENV_TWO
  restartPolicy: Never


Deploy:
---------
kubectl apply -f cm-pod-arg.yaml

Validate:
---------
kubectl logs cm-pod-arg
```

Injecting ConfigMap into As Files inside Volume(3/3)

```
# cm-pod-file-vol.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cm-pod-file-vol
spec:
  volumes:
    - name: mapvol
      configMap:
        name: my-nginx-config-yaml
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
      - name: mapvol
        mountPath: /etc/config
  restartPolicy: Never
```

Deploy:
----
kubectl apply -f cm-pod-file-vol.yaml

Validate:
----
kubectl exec configmap-vol-pod -- ls /etc/config
kubectl exec configmap-vol-pod -- cat /etc/config/etc/config/my-nginx-config.conf

Running operations directly on the YAML file:
~~~~~~

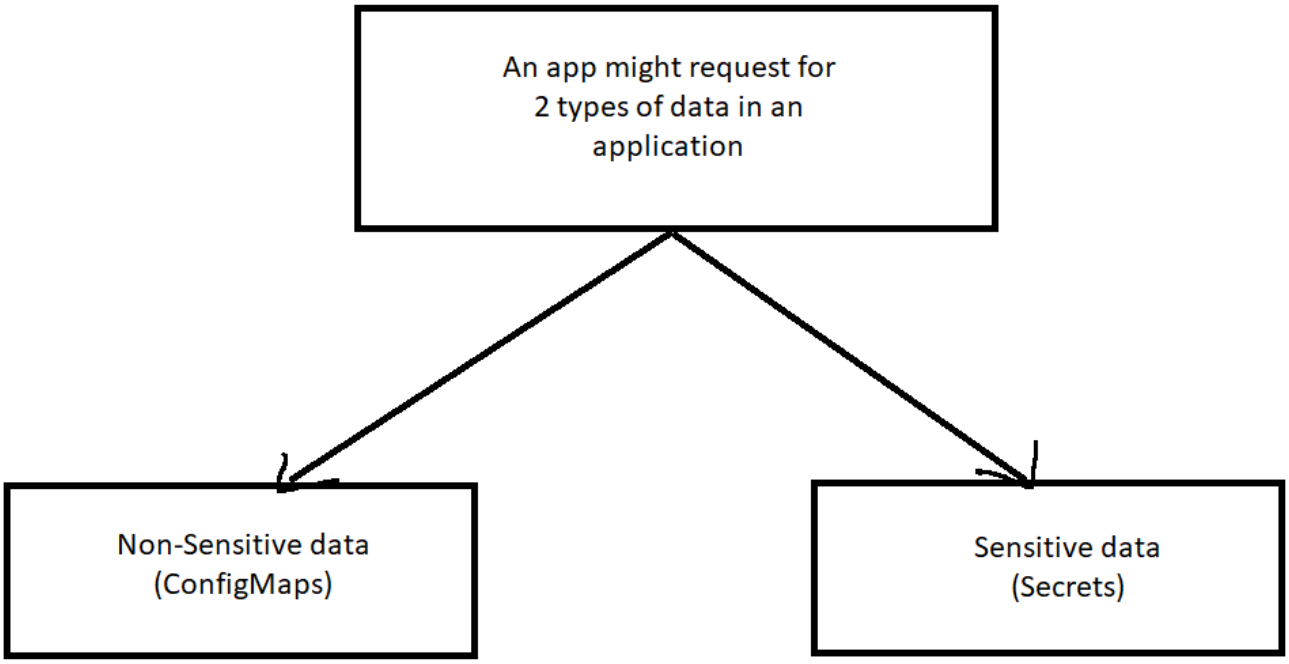
kubectl [OPERATION] –f [FILE-NAME.yaml]
kubectl get –f [FILE-NAME.yaml]
kubectl delete –f [FILE-NAME.yaml]
kubectl get -f [FILE-NAME.yaml]
kubectl create -f [FILE-NAME.yaml]

Delete ConfigMap:
~~~~~~~~~~~~~~~~~~~~~~

kubectl delete configmap <NAME>

31. Secrets

Wednesday, June 7, 2023 10:56 AM



32. Secrets - Tasks

Wednesday, June 7, 2023 10:59 AM

Creating Secrets Declaratively (Using YAML):

~~~~~~

Base-64 Encoding:

---

echo -n 'admin' | base64
echo -n 'pas@word1' | base64

// copy its output

****

Using Base64 Encoding in creating Secret:

```
# cat db-user-pass.yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-user-pass
  namespace: default
data:
  username: <o/p of admin>
  password:  <o/p of pwd>
```

Deploy:

---

kubectl apply –f secret-db-user-pass.yaml

```
[root@kmaster1 ~]$  echo -n 'admin' | base64
YWRtaW4=
[root@kmaster1 ~]$  echo -n 'pecho -n 'pas@word1' | base64
cGFzQHdvcmQx
[root@kmaster1 ~]$  vim db-user.yaml
[root@kmaster1 ~]$  kubectl apply -f db-user.yaml
secret/db-user-pass created
[root@kmaster1 ~]$
[root@kmaster1 ~]$  kubectl get secrets
NAME         TYPE     DATA   AGE
```
~~~~~~

```
db-user-pass   Opaque   2     30s
[root@kmaster1 ~]$
[root@kmaster1 ~]$  kubectl describe  secrets  db-user-pass
Name:        db-user-pass
Namespace:    default
Labels:      <none>
Annotations:  <none>

Type:  Opaque

Data
====
username:  5 bytes
password:  9 bytes
[root@kmaster1 ~]$
```

****

Creating Secrets Imperatively (From Command line)

kubectl create secret generic test-secret --from-literal='username=my-app' --from-literal='password=39528$vdg7Jb'

Creating Secrets Imperatively (From files):

echo -n 'admin' > ./username.txt
echo -n '1f2d1e2e67df' > ./password.txt

kubectl create secret generic db-user-pass-from-file --from-file=./username.txt --from-file=./password.txt

Example:
---
kubectl get secrets db-user-pass –o yaml

Injecting "Secrets" into Pod As Environmental Variables:

```
# my-secrets-pod-env.yaml

apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
```

```
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: db-user-pass
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: db-user-pass
            key: password
  restartPolicy: Never
```

Validate:

```
kubectl exec secret-env-pod -- env
kubectl exec secret-env-pod -- env | grep SECRET
```

****

Injecting "Secrets" into Pod As Files inside the Volume:

```
# cat my-secrets-vol-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-vol-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: secret-volume
          mountPath: /etc/secret-volume
  volumes:
```

```
  - name: secret-volume
    secret:
      secretName: test-secret
```

Validate:

---

kubectl exec secret-vol-pod -- ls /etc/secret-volume
kubectl exec secret-vol-pod -- cat /etc/secret-volume/username
kubectl exec secret-vol-pod -- cat /etc/secret-volume/password

****

Displaying Secret:

~~~~~~~~~~~~~~~~~~~~~~

kubectl get secret <NAME>
kubectl get secret <NAME> -o wide
kubectl get secret <NAME> -o yaml
kubectl get secret <NAME> -o json
kubectl describe secret <NAME>

Running operations directly on the YAML file:

~~~~~~

kubectl [OPERATION] –f [FILE-NAME.yaml]

kubectl get –f [FILE-NAME.yaml]
kubectl delete –f [FILE-NAME.yaml]
kubectl get -f [FILE-NAME.yaml]
kubectl create -f [FILE-NAME.yaml]

****

Delete Secret:

~~~~~~~~~~~~~~~~~~

kubectl delete secret <NAME>
~~~~~~~~~~~~~~~~~~

33. NodeSelector

Thursday, June 8, 2023 11:10 AM

1. Labeling Node:

---

kubectl get nodes --show-labels
kubectl label nodes worker-1 disktype=ssd

kubectl get nodes --show-labels
kubectl get pods -o wide

****
****

2. Deploying Node-Selector YAML:

```
# nodeSelector-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nodeselector-pod
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

---

kubectl apply -f ns.yaml

****
****

3. Testing:

---

kubectl get pods -o wide
kubectl get nodes --show-labels

Let's Delete and Deploy "again" to ensure Pod is deployed on the same node which is lablelled above.
kubectl delete -f ns.yaml
kubectl apply -f ns.yaml

kubectl get pods -o wide
kubectl get nodes --show-labels

****
****

4. Cleanup:

---

kubectl label nodes worker-1 disktype-
kubectl delete pods nodeselector-pod

****
****

34. Metrics Server & Resource Request & Limit

Thursday, June 8, 2023 11:20 AM

Step 1) Download Metrics Server Manifest

curl -LO https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Step 2) Modify Metrics Server Yaml File

vi components.yaml

under spec, add:

hostNetwork: true

goto: 140

- --kubelet-insecure-tls

```
spec:
  selector:
    matchLabels:
      k8s-app: metrics-server
  strategy:
    rollingUpdate:
      maxUnavailable: 0
  template:
    metadata:
      labels:
        k8s-app: metrics-server
    spec:
      hostNetwork: true
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: registry.k8s.io/metrics-server/metrics-server:v0.6.3
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /livez
            port: https
            scheme: HTTPS
          periodSeconds: 10
        name: metrics-server
```

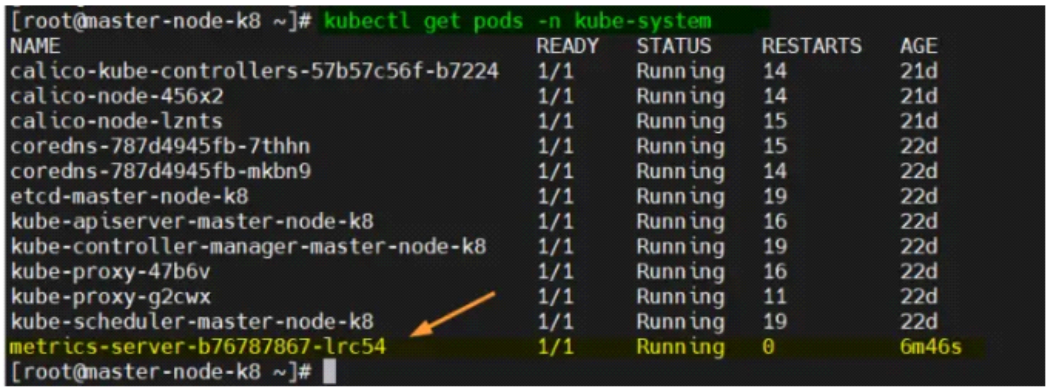
Step 3) Deploy Metrics Server

kubectl apply -f components.yaml

```
[root@master-node-k8 ~]# kubectl apply -f components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
[root@master-node-k8 ~]#
```

Step 4) Verify Metrics Server Deployment

kubectl get pods -n kube-system



Step 5) Test Metrics Server Installation
kubectl top nodes

kubectl top pod
kubectl top pod -n kube-system

##
1. Configuring Container with "Memory" Requests and Limits:
~~~~~~

a. First, get the output of kubectl top command to find resources that are "currently consumed".
kubectl top nodes

==========
==========

b. Then, deploy this Pod with memory requests and limits as mentioned below
---
#memory-demo.yaml
apiVersion: v1
kind: Pod
metadata:
 name: memory-demo
spec:
 containers:
 - name: memory-demo-ctr
  image: polinux/stress
  resources:
   requests:
    memory: "100Mi"
   limits:
    memory: "200Mi"
  command: ["stress"]
  args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

==========
==========
~~~~~~

c. Deploy:
---
kubectl apply -f memory-demo.yaml

==========

d. Validate
---

kubectl get pods -o wide

kubectl top nodes

****

35. Self-healing / auto-healing

Thursday, June 8, 2023 11:12 AM

- A system that heals itself without any manual intervention even in an undesirable situation.
- K8s is self-healing system.
- K8s provides multiple solutions to heal itself. Like:
    - Replicaset
    - Deployment
    - Daemon set
        - Etc.

- **Replication**:
    - Kubernetes allows you to define the desired number of replicas for your application.
    - If a pod (an instance of an application) fails, Kubernetes automatically creates new replicas to maintain the desired state.
- **Health checks**:
    - Kubernetes performs health checks on pods using readiness and liveness probes.
    - Readiness probes determine if a pod is ready to serve traffic, while liveness probes check if a pod is still running.
    - If a pod fails these checks, Kubernetes automatically restarts or recreates the pod.
- **Rolling updates**:
    - When deploying new versions of applications, Kubernetes supports rolling updates.
    - It gradually replaces the old instances of the application with the new ones, ensuring minimal downtime and maintaining the desired state throughout the update process.
- **Pod rescheduling**:
    - If a node in the cluster fails or becomes unresponsive, Kubernetes detects it and automatically reschedules the affected pods to other healthy nodes.
    - This ensures that the application continues running even in the face of node failures.
- **Daemonsets**:
    - Kubernetes Daemonsets are used for running system-level daemons on every node in the cluster.
    - If a node is added or removed from the cluster, Daemonsets automatically manage the deployment or termination of daemons to maintain the desired state.
- **Self-monitoring**:
    - Kubernetes itself is self-monitoring, continuously checking the health and state of the cluster components.
    - If it detects any issues, Kubernetes attempts to resolve them automatically or alerts the cluster administrators.

36. Manifest & templating

Thursday, June 8, 2023 11:12 AM

1. Helm
2. Kustomize

Refer to handbook

37. Networking in K8s

Thursday, June 8, 2023 11:18 AM

Refer to NB

38. Monitoring Nodes CPU & RAM - tested

Thursday, June 8, 2023 11:49 AM

Step 1) Download Metrics Server Manifest

curl -LO https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Step 2) Modify Metrics Server Yaml File

vi components.yaml

under spec, add:

hostNetwork: true

goto: 140

- --kubelet-insecure-tls

```
spec:
  selector:
    matchLabels:
      k8s-app: metrics-server
  strategy:
    rollingUpdate:
      maxUnavailable: 0
  template:
    metadata:
      labels:
        k8s-app: metrics-server
    spec:
      hostNetwork: true
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: registry.k8s.io/metrics-server/metrics-server:v0.6.3
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /livez
            port: https
            scheme: HTTPS
          periodSeconds: 10
        name: metrics-server
```

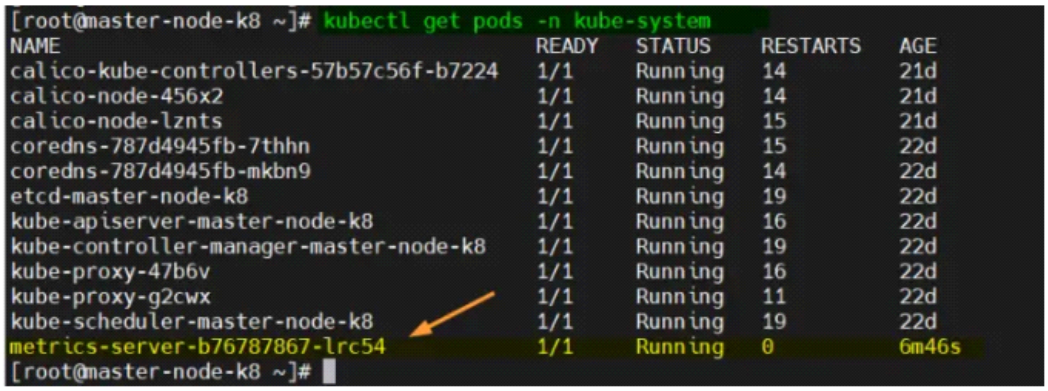
Step 3) Deploy Metrics Server

kubectl apply -f components.yaml

```
[root@master-node-k8 ~]# kubectl apply -f components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
[root@master-node-k8 ~]#
```

Step 4) Verify Metrics Server Deployment

kubectl get pods -n kube-system



Step 5) Test Metrics Server Installation

kubectl top nodes

kubectl top pod

kubectl top pod -n kube-system

39. Cluster Networking

Wednesday, June 14, 2023 6:28 AM

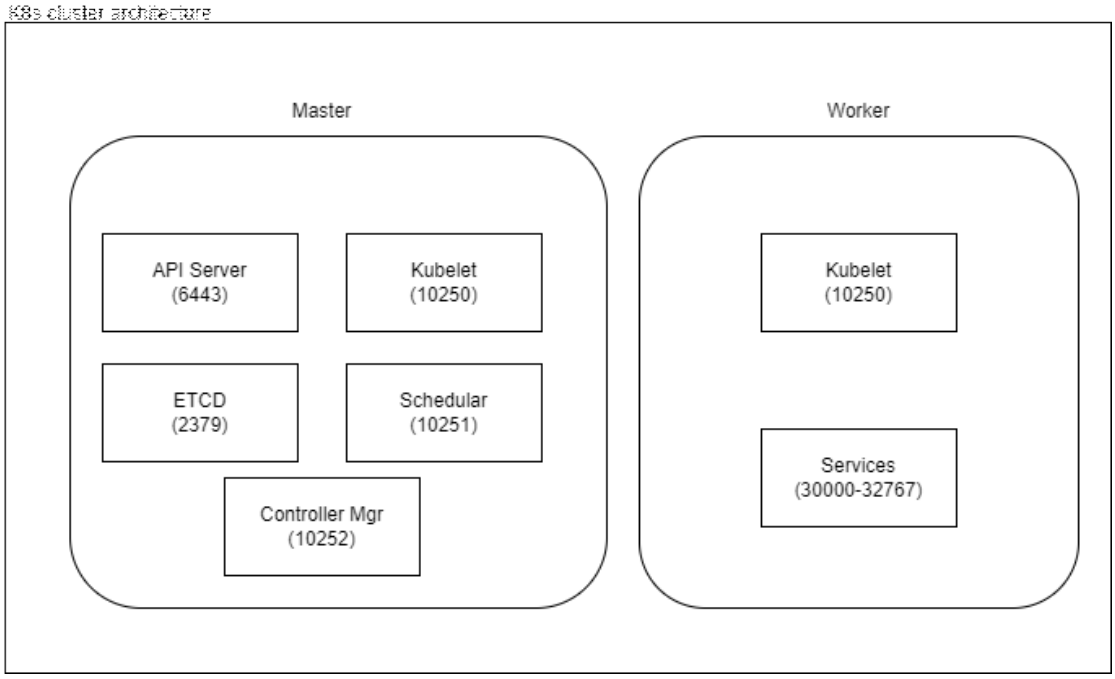
K8s network fundamental requirements:

a. All pods can communicate with all pods without using NAT.
b. All nodes can communicate with all pods without NAT.
c. The IP that a pod sees itself as is the same IP that allows others sees it as.

CNI-plugins

1. Calico
2. Flannel
3. Weave
4. Romana

Networking ports:

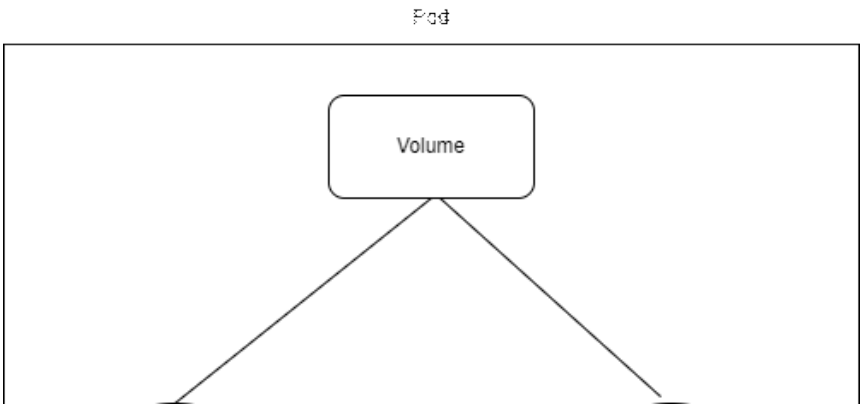


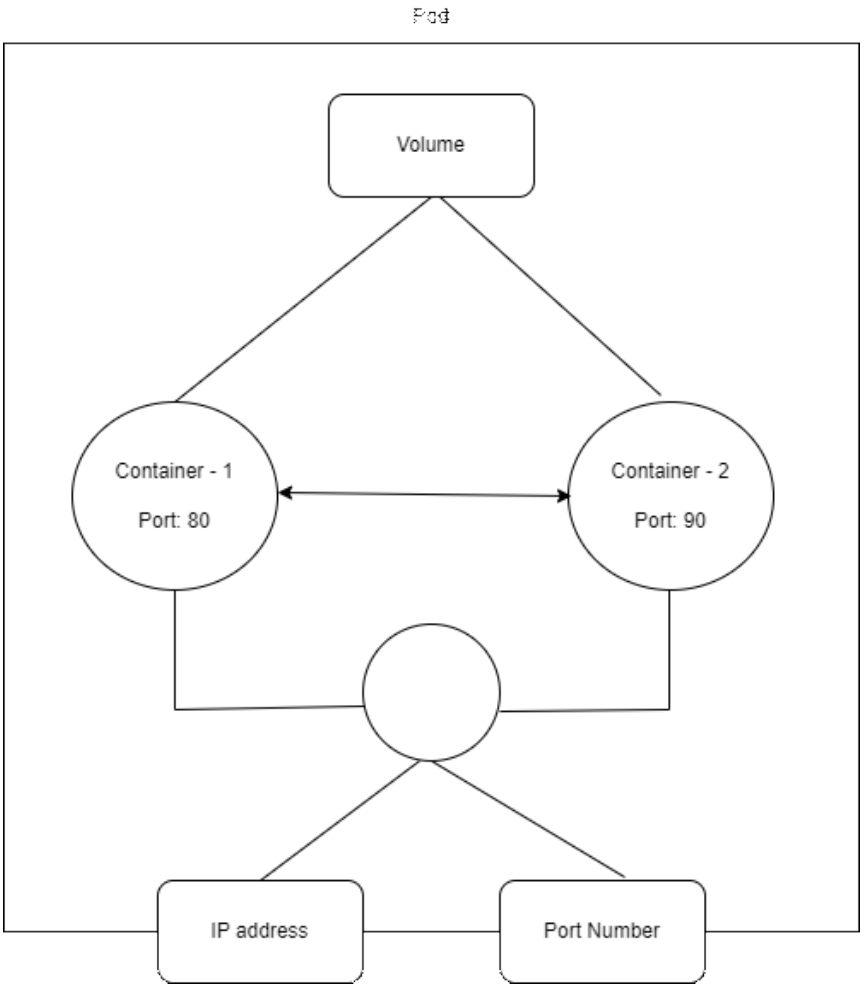
There are several communication methods:

1. Container to container
2. Pod to pod
3. Pod to service
4. External to service

Container to container

- Aka intra-pod communication.





From container 1, you want to access container 2
curl localhost:80

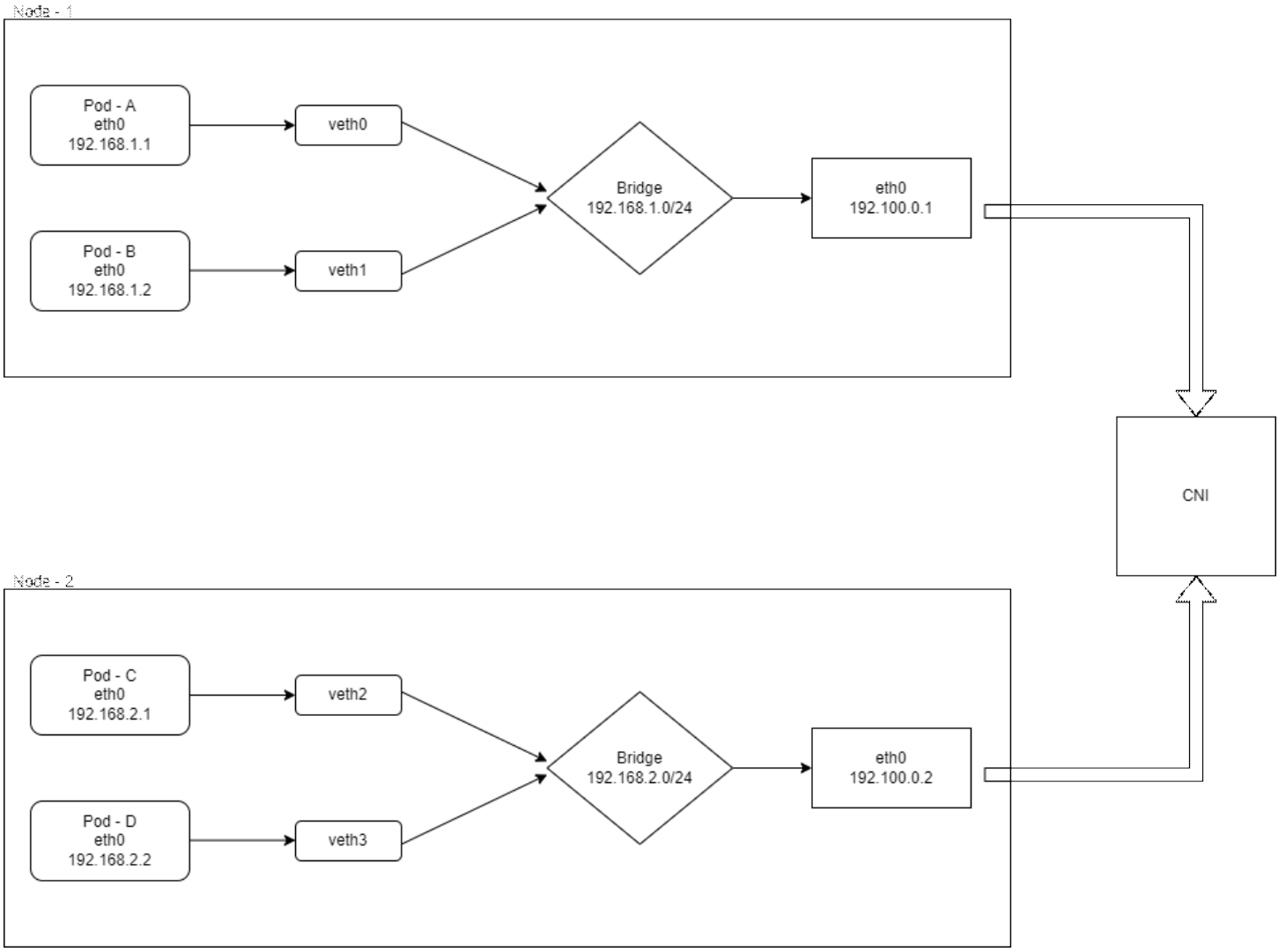
Vice-versa,
curl localhost:90

Creating intra-pod communication (YAML):

```
[root@kmaster intra-pod]$  cat intra-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: intra-pod-example
spec:
  containers:
  - name: webserver
    image: nginx
  - name: centos
    image: centos
    command: ["/bin/sh","-c","while : ; do curl http://localhost:80/; sleep 10; done"]
```

```
[root@kmaster intra-pod]$  cat intra-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: intra-pod-example
spec:
  containers:
  - name: webserver
    image: nginx
  - name: centos
    image: centos
    command: ["/bin/sh","-c","while : ; do curl http://localhost:80/; sleep 10; done"]
```

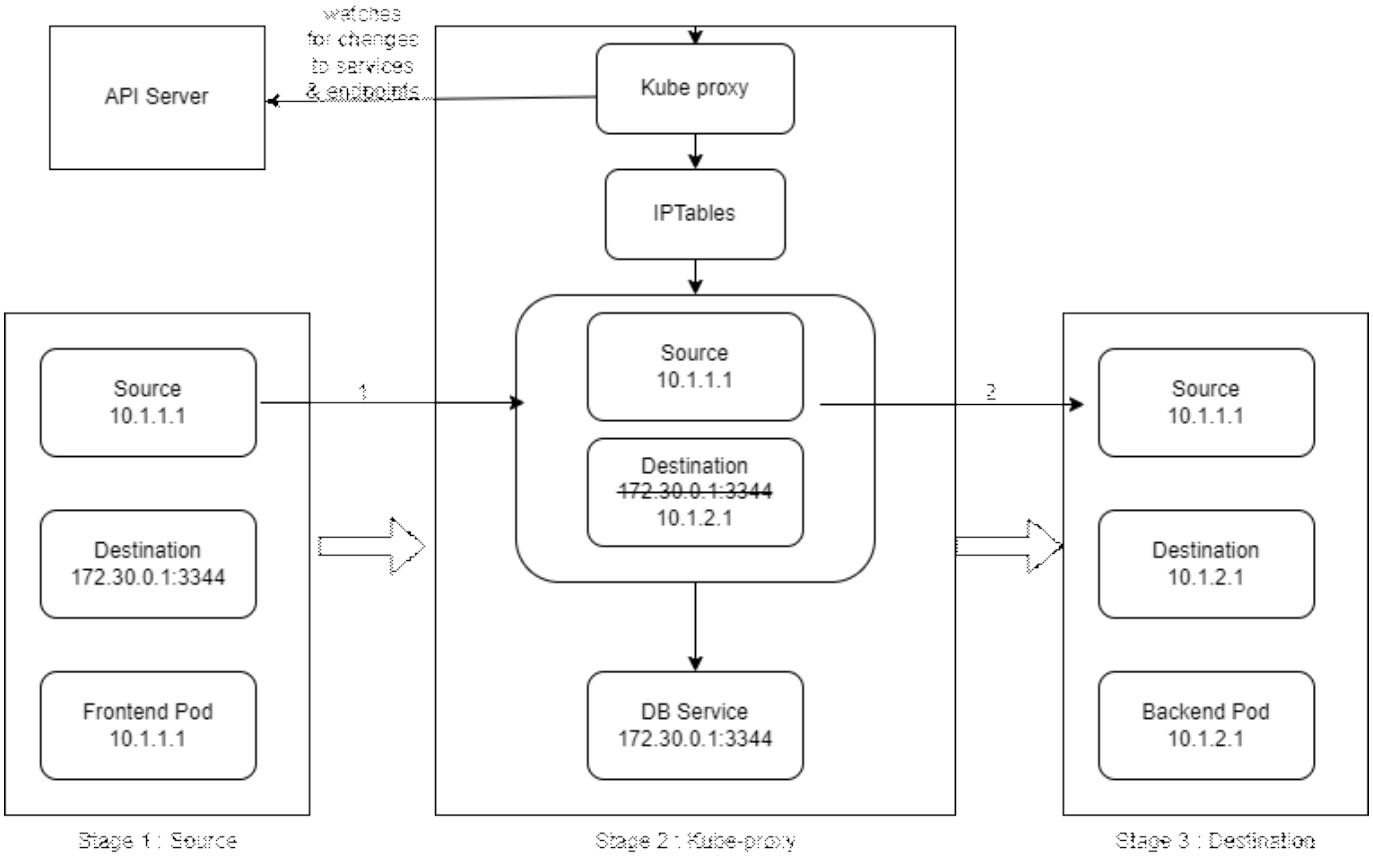
Pod to pod communication:



If Pod-A wants to talk to Pod-B (communication between pods)
Pod-A --> veth0 --> bridge --> veth1 --> pod-B

If pod-B wants to talk to pod-D (communication between pods, across nodes)
Pod-B --> eth1 --> veth1 --> bridge --> CNI --> bridge --> veth3 --> pod-D

Pod to service communication:



Internet to service

To expose applications on internet, we have 3 ways

1. Node Port service
2. Load balancer
3. Ingress (it's not a svc, but a resource)

40. NodePort service

Wednesday, June 14, 2023 8:17 AM

##

GCP -> search for firewall rules & create 2 rules.

1. For Master Node:

- target - "All inst in the n/w"
- source filter: IPv4 ranges
- source range: 0.0.0.0/0
- specified protocols & ports.

TCP --> 2379, 6443, 10250, 10251, 10252

same for worker node

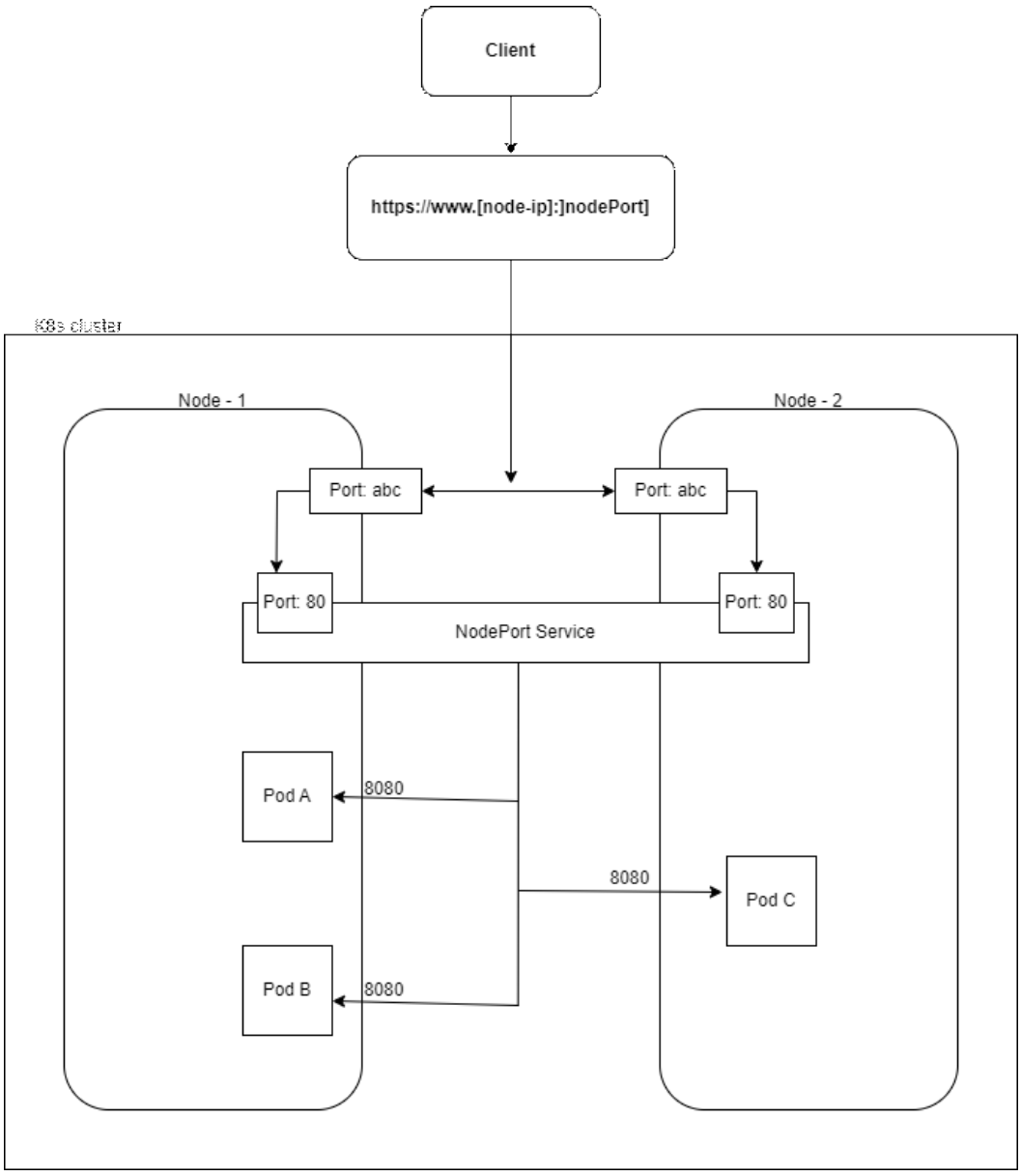
1. For Worker Nodes: TCP Ingress with Ports - 10250, 30000-32767

##

- It opens port on internet.
- Good for demos & testing purpose.

Disadvantage:

- Whole app is accessible on a single port over internet.
- When node fails, then no service will be present.



NodePort Yaml config file:
~~~~~~~~~~~~~~~~~~~~~~~~

Creating NP service:

```
[root@kmaster nodeport]$  cat nginx-svc-np.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
    app: nginx-app
spec:
  selector:
    app: nginx-app
  type: NodePort
  ports:
  - nodePort: 31111
    port: 80
    targetPort: 80
```

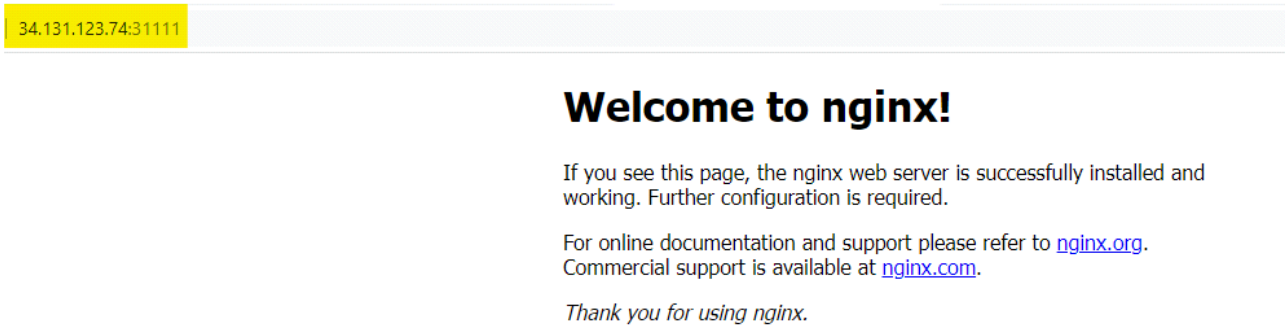
Deploying the config

```
[root@kmaster nodeport]$  cat nginx-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.18
        ports:
        - containerPort: 80
```

Fetching the pod info:

```
[root@kmaster nodeport]$  kubectl get pod -o wide
NAME                          READY   STATUS    RESTARTS   AGE    IP         NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-85d56c7d7d-q2f8h   1/1     Running   0          2m21s   10.47.0.1   worker-2   <none>           <none>

[root@kmaster nodeport]$  kubectl get svc
NAME         TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
my-service   NodePort   10.98.143.80   <none>        80:31111/TCP   3m33s
```

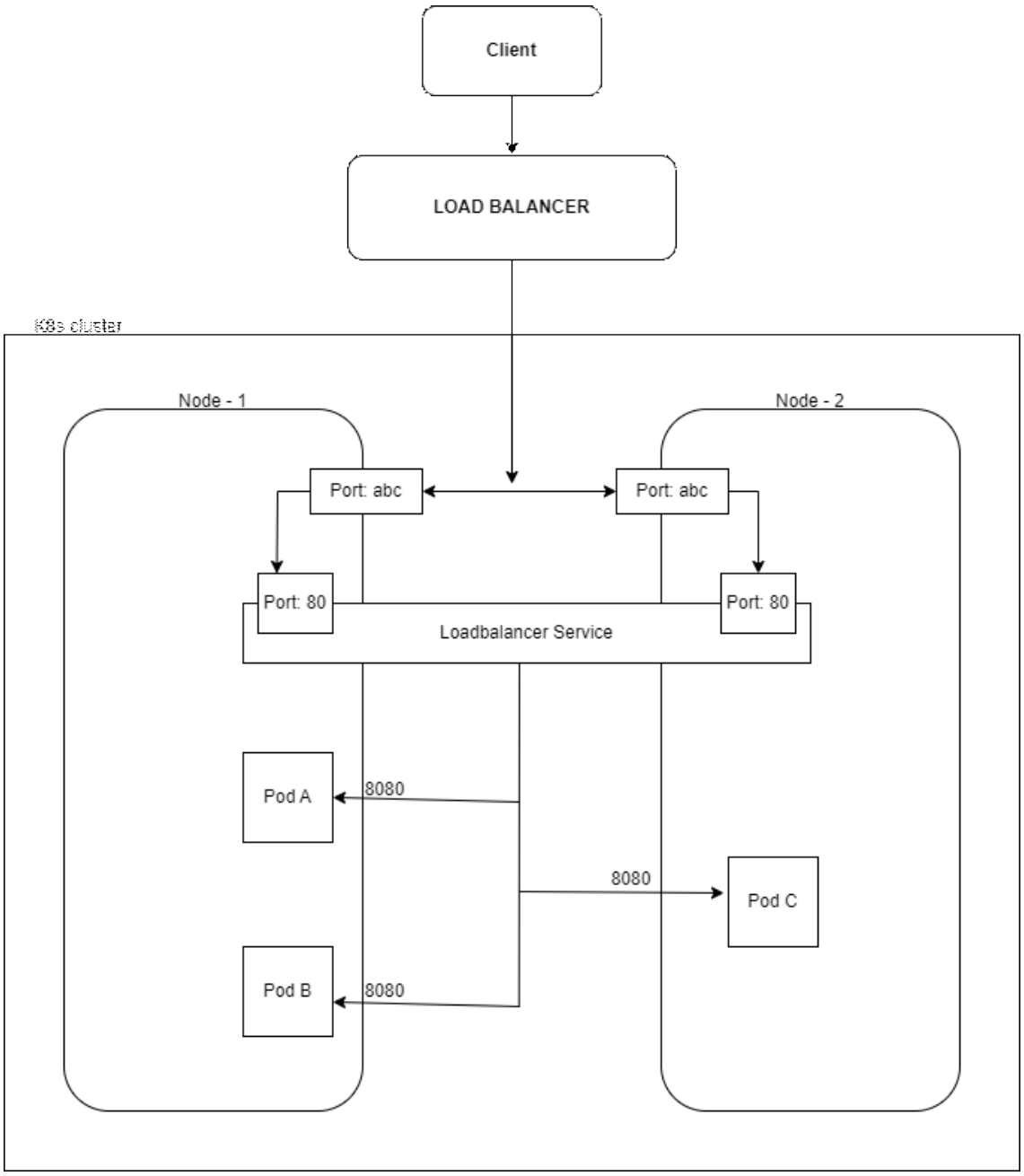
Accessing the website on port worker-2-IP-address:31111:
~~~~~~~~~~~~~~~~~~~~~~~~


34.131.123.74:31111
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

41. Load balancer service

Wednesday, June 14, 2023 8:48 AM

- Adv: good for exposing a service directly on internet.
- Disadv: LB is expensive, multiple services == multiple LBs.



YAML config file:

```
[root@kmaster ~]$  cat loadbalancer.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
```

```
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-container
          image: nginx:latest
          ports:
            - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalancer
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport
spec:
  type: NodePort
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 31111
      targetPort: 80
      nodePort: 31111

[root@kmaster ~]$  kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/my-deployment-544587cc57-57zgg   1/1     Running   0          16s
pod/my-deployment-544587cc57-bmw48   1/1     Running   0          16s
pod/my-deployment-544587cc57-rtxsm   1/1     Running   0          16s

NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes       ClusterIP      10.96.0.1       <none>        443/TCP          20d
service/my-loadbalancer   LoadBalancer   10.105.51.221   <pending>     80:32766/TCP     16s
service/my-nodeport      NodePort       10.105.86.185   <none>        31111:31111/TCP   16s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-deployment   3/3     3            3           16s
```

```
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/my-deployment-544587cc57   3         3         3       16s

[root@kmaster ~]$  kubectl get pods -o wide
NAME                             READY   STATUS    RESTARTS   AGE   IP          NODE       NOMINATED NODE   READINESS GATES
my-deployment-544587cc57-57zgg   1/1     Running   0          40s   10.47.0.2   worker-2   <none>           <none>
my-deployment-544587cc57-bmw48   1/1     Running   0          40s   10.32.0.4   worker-1   <none>           <none>
my-deployment-544587cc57-rtxsm   1/1     Running   0          40s   10.47.0.1   worker-2   <none>           <none>
```

copy the GCP VM's public IP and fetch the port number from "kubectl get all" cmd
Ex: http://wrk-1-IP-address:32766/

deleting the resources:
kubectl delete deploy my-deployment
kubectl get all
kubectl delete svc my-loadbalancer my-nodeport --force

42. HostPath volumes

Wednesday, June 14, 2023 10:09 AM

HostPath volume
~~~~~~~~~~~~~~~~~

 # mkdir /test-vol

 # cat > /test-vol/dummy.txt
this is Jeetu.

 # cat nginx-hostpath.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-hostpath
spec:
  containers:
    - name: nginx-container
      image: nginx
      volumeMounts:
      - mountPath: /test-mnt
        name: test-vol
  volumes:
  - name: test-vol
    hostPath:
      path: /test-vol
```

Deploy:
---
kubectl apply -f nginx-hostpath.yaml

get inside pod:
kubectl exec -it nginx-hostpath -- /bin/bash

list mount points:
df -h
~~~~~~~~~~~~~~~~~

list contents:

```
# ls -h /test-mnt/
```

43. Persistent storage

Wednesday, June 14, 2023 10:37 AM

1. On Cluster Configured with Kubeadm

---

a. First, login and created the disk.

```
# gcloud auth login
# gcloud compute disks create --size=10GB --zone=us-central1-a my-data-disk-1
```

b. Next, create the Pod YAML.

```
# gce-pd.yaml
apiVersion: v1
kind: Pod
metadata:
  name: gce-pd
spec:
  containers:
  - image: mongo
    name: mongodb
    volumeMounts:
    - name: mongodb-data
      mountPath: /data/db
  volumes:
  - name: mongodb-data
    gcePersistentDisk:
      pdName: my-data-disk-1
      fsType: ext4
```

c. Deploy.

```
kubectl apply -f gce-pd.yaml
```

d. Find, the worker node which this Pod is deployed.

---

kubectl get pods -o wide

e. Attach the disk to respective "worker" node from Google Cloud Dashboad.

---

f. login to the respective "worker" node and run following commands

---

mkfs.ext4 /dev/disk/by-id/scsi-0Google_PersistentDisk_my-data-disk-1

mkdir -p /var/lib/kubelet/plugins/kubernetes.io/gce-pd/mounts/my-data-disk-2 && mount /dev/disk/by-id/scsi-0Google_PersistentDisk_my-data-disk-2 /var/lib/kubelet/plugins/kubernetes.io/gce-pd/mounts/my-data-disk-2

g. Wait few mins. Then, login to the "master" node and display Pods and it status to ensure it is "Running"

---

kubectl get pods

h. Validate:

---

kubectl exec gce-pd -it -- df /data/db

THEN ALSO ITS NOT MANDATORY, THAT IT WILL WORK. DUE TO THIS WE HAVE PV & PVC IN K8S.

****
****

2. On GKE:
---

a. First, create the disk.
---
gcloud compute disks create --size=10GB --zone=us-central1-c my-data-disk-2

b. Next, create the Pod.

Use above file

c. Deploy.
---
kubectl apply -f gce-pd.yaml

****
****

Cleanup:
---

kubectl delete pods gce-pd

gcloud compute disks delete --zone=us-central1-a my-data-disk-1

gcloud compute disks delete --zone=us-central1-c my-data-disk-2

44. PersistentVolume (PV) & PersistentVolumeClaim (PVC)

Wednesday, June 14, 2023 10:42 AM

PersistentVolume (PV)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
[root@kmaster ~]$  cat pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: manual
  hostPath:
    path: /mnt/data
[root@kmaster ~]$
```

List for any PVs:

```
# kc get pv
```

Apply the config:

```
# kc apply -f pv.yaml
```

List for any PVs: --> STATUS is AVAILABLE

```
# kc get pv
```

PersistentVolumeClaim (PVC)

~~~~~~

List PVC

```
# kubectl get pvc
```

```
[root@kmaster ~]$  cat pvc.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
[root@kmaster ~]$
```

Applying the confg:

kc apply -f pvc.yaml

Checking PVC status: --> STATUS is BOUND & default/my-pvc

kc get pvc

Attaching this PVC to the pod:

~~~~~~~~~~~~~~~~~~~~~~~~~~

```
[root@kmaster storage]$  cat pvc-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: nginx
      volumeMounts:
        - name: my-volume
          mountPath: /data
  volumes:
    - name: my-volume
```
~~~~~~~~~~~~~~~~~~~~~~~~~~

```
    persistentVolumeClaim:
      claimName: my-pvc
```

apply

```
# kubectl apply -f pvc-pod.yaml
```

Verify

```
# kubectl exec my-pod -it -- /bin/bash
root@my-pod:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         9.6G  6.1G  3.5G  64% /
tmpfs            64M     0   64M   0% /dev
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/sda1       9.6G  6.1G  3.5G  64% /data
shm              64M     0   64M   0% /dev/shm
tmpfs           3.8G   12K  3.8G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs           2.0G     0  2.0G   0% /proc/acpi
tmpfs           2.0G     0  2.0G   0% /proc/scsi
tmpfs           2.0G     0  2.0G   0% /sys/firmware

root@my-pod:/# ls /data
my-file
root@my-pod:/# cat /data/my-file
hello
```

Deletion

1. Delete pod - # kubectl delete pod my-pod
2. Delete PVC - # kubectl delete pvc my-pvc
3. Delete PV - # kubectl delete pv my-pv

45. StorageClass

Wednesday, June 14, 2023 10:55 AM

SC YAML code:
~~~~~~~~~~~~~~~

```
[root@kmaster ~]$  cat sc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
  reclaimPolicy: Retain
```

Checking for any SC:
kc get sc

Applying the manifest file:
kc apply -f sc.yaml

Checking for SC again:
kc get sc

```
NAME      PROVISIONER            RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
standard  kubernetes.io/aws-ebs   Delete          Immediate           false                  8s
```
~~~~~~~~~~~~~~~

46. Monitoring

Wednesday, June 14, 2023 10:31 PM

- Tracking all/any resources for efficiently working of cluster.
- Why monitoring?
  1. Reliability
  2. Insights
  3. Cost control

PRE-REQ: Installing Metrics-Server (Required for "kubectl top" command)
~~~~~~

a. Download
---
git clone https://github.com/kubernetes-sigs/metrics-server.git

b. Install
---
kubectl apply -k metrics-server/manifests/test

c. Validate
---
kubectl get deployment metrics-server -n kube-system
kubectl get pods -n kube-system | grep metrics
kubectl get apiservices | grep metrics
kubectl top pods
kubectl top nodes

Note: Give it a minute if nothing shows up in top command output.

1. NODEs: Ensure all nodes are "Healthy" and "Ready". Monitor its resource usage (CPU & Memory):
~~~~~~

kubectl run nginx-pod --image=nginx
kubectl run redis-pod --image=redis

kubectl top nodes
kubectl top nodes --sorty-by cpu
kubectl top nodes --sorty-by memory
kubectl top nodes --sorty-by memory > mem-out.txt

****

2. PODs: Ensure all Pods are "Running" successfully and Monitor its resource usage (CPU & Memory):
~~~~~~

In Current Namepsace:
---
kubectl top pods
kubectl top pods --sorty-by cpu
kubectl top pods --sorty-by memory
kubectl top pods --sorty-by memory > mem-out.txt

In Specific NameSpace:
---
kubectl top pods -n [NAME-SPACE]
kubectl top pods -n [NAME-SPACE] --sorty-by cpu
kubectl top pods -n [NAME-SPACE] --sorty-by memory
kubectl top pods -n [NAME-SPACE] --sorty-by memory > mem-out.txt

In Across Namespaces:
---
kubectl top pods -A
kubectl top pods -A --sorty-by cpu
kubectl top pods -A --sorty-by memory
kubectl top pods -A --sorty-by memory > mem-out.txt

When Multi-Cotinaer Pod:
---
kubectl top pod [POD-NAME] --containers

****

3. Cluster Compoents: Ensure all K8s Cluster Components are "Healthy" and "Running" status:
~~~~~~

If Cluster configured with "Kubeadm"
---
kubectl get pods -n kube-system

If cluster configured Manually (the hard-way)
---
systemctl status kube-apiserver
systemctl status kube-controller-manager
systemctl status kube-scheduler

Ensure, following components are in "Running" status on all nodes including Master(Control-Plane)

node.

---

systemctl status docker
systemctl status kubelet

47. Logging

Thursday, June 15, 2023 10:04 AM

```
# counter.yaml
apiVersion: v1
kind: Pod
metadata:
  name: one-counter-pod
spec:
  containers:
  - name: counter-container
    image: busybox
    args: [/bin/sh, -c,
          'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']

---

apiVersion: v1
kind: Pod
metadata:
  name: two-counter-pod
spec:
  containers:
  - name: counter-1
    image: busybox
    args: [/bin/sh, -c,
          'i=0; while true; do echo "From Counter-ONE: $i: $(date)"; i=$((i+1)); sleep 1; done']
  - name: counter-2
    image: busybox
    args: [/bin/sh, -c,
          'i=0; while true; do echo "From Counter-TWO:  $i: $(date)"; i=$((i+1)); sleep 1; done']
```

****

2. Display Container Logs using "kubectl logs" command:

```
kubectl logs [POD-NAME]                    # dump pod logs (stdout)

kubectl logs -f [POD-NAME]                 # stream pod logs (stdout)

kubectl logs [POD-NAME] –-since=5m         # view logs for last 5 mins (h for hours)

kubectl logs [POD-NAME] --tail=20          # Display only more recent 20 lines of output in pod
```

kubectl logs [POD-NAME] --previous # dump pod logs (stdout) for a previous instantiation of a container

kubectl logs [POD-NAME] > [FILE-NAME].log # Save log output to a file

kubectl logs [POD-NAME] -c [CONTAINER-NAME] # dump pod container logs (stdout, multi-container case)

kubectl logs [POD-NAME] --all-containers=true # dump logs of all containers inside nginx pod

kubectl logs -l [KEY]=[VALUE] # dump pod logs, with label (stdout)

****

3. Using Journalctl:
---

journalctl #Display all messages

journalctl -r #Display newest log entries first (Latest to Old order)

journalctl -f #Enable follow mode & display new messages as they come in

journalctl –n 3 #Display specific number of RECENT log entries

journalctl –p crit #Display specific priority – “info”, “warning”, “err”, “crit”, “alert”, “emerg”

journalctl –u docker #Display log entries of only specific systemd unit

journalctl –o verbose #Format output in “verbose”, “short”, “json” and more

journalctl –n 3 –p crit #Combining options

journalctl --since "2019-02-02 20:30:00" --until "2019-03-31 12:00:00" # Display all messages between specific duration

****

4. Display Container Logs using "docker logs" command from respective worker node:
---

kubectl get pods -o wide

docker ps | grep [KEY-WORD]

docker logs [CONTAINER-ID]

****
****

5. K8s Cluster Component Logs:
---
journalctl –u docker
journalctl –u kubelet

If K8s Cluster Configured using "kubeadm":
---
kubectl logs kube-apiserver-master -n kube-system | more
kubectl logs kube-controller-manager-master -n kube-system
kubectl logs kube-scheduler-master -n kube-system
kubectl logs etcd-master -n kube-system

If K8s Cluster Configured using "Hard-way (Manual)"
---
journalctl –u kube-apiserver
journalctl –u kube-scheduler
journalctl –u etcd
journalctl –u kube-controller-manager

0.Pre-requisites

Friday, May 19, 2023 4:29 PM

- Install Terraform
  - Set the environment variables.
- Install Azure CLI
  - Link: https://aka.ms/installazurecliwindows
- Install VS Code editor
  - Link: https://code.visualstudio.com/download
    Install extension
    - Azure CLI
    - Terraform (by hashicorp)

Execute below command on VS code editor terminal
az login

1. list of commands

Friday, May 19, 2023 4:33 PM

List of commands:

Check terraform version:	#terraform version
Check AZ version:	#az version
Initiating terraform env:	#terraform init
Viewing/Planning terraform:	#terraform plan
Applying terraform code in Azure:	#terraform apply
Destroying terraform:	#terraform destroy

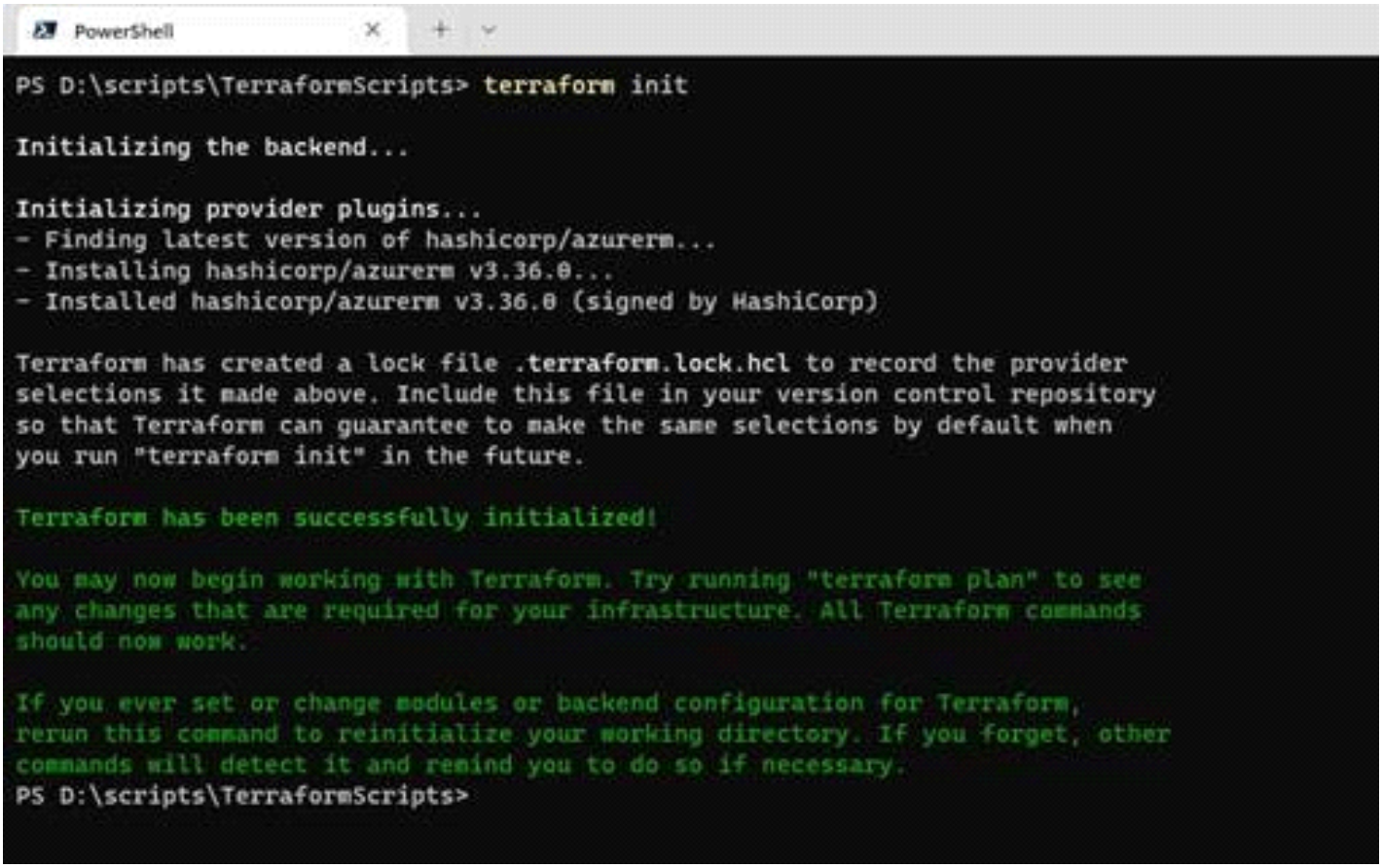
Creating Service Principal for Azure:
#az ad sp create-for-rbac --name="SPForTerraform" --role="Contributor" --scopes="/subscriptions/6923f8cc-4638-4832-a0e7-63be3ca5c15b"

Creds:
subscription_id = "6923f8cc-4638-4832-a0e7-63be3ca5c15b"
client_id = "436ec27b-6069-4bb6-a872-1376f17a4d27"
client_secret = "kgt8Q~y2meiwek8YtMKoX64rv3qJ8b7ER0Y0Fbkq"
tenant_id = "5659fac0-8e34-40af-86b2-dfcd9b0ddbf3"

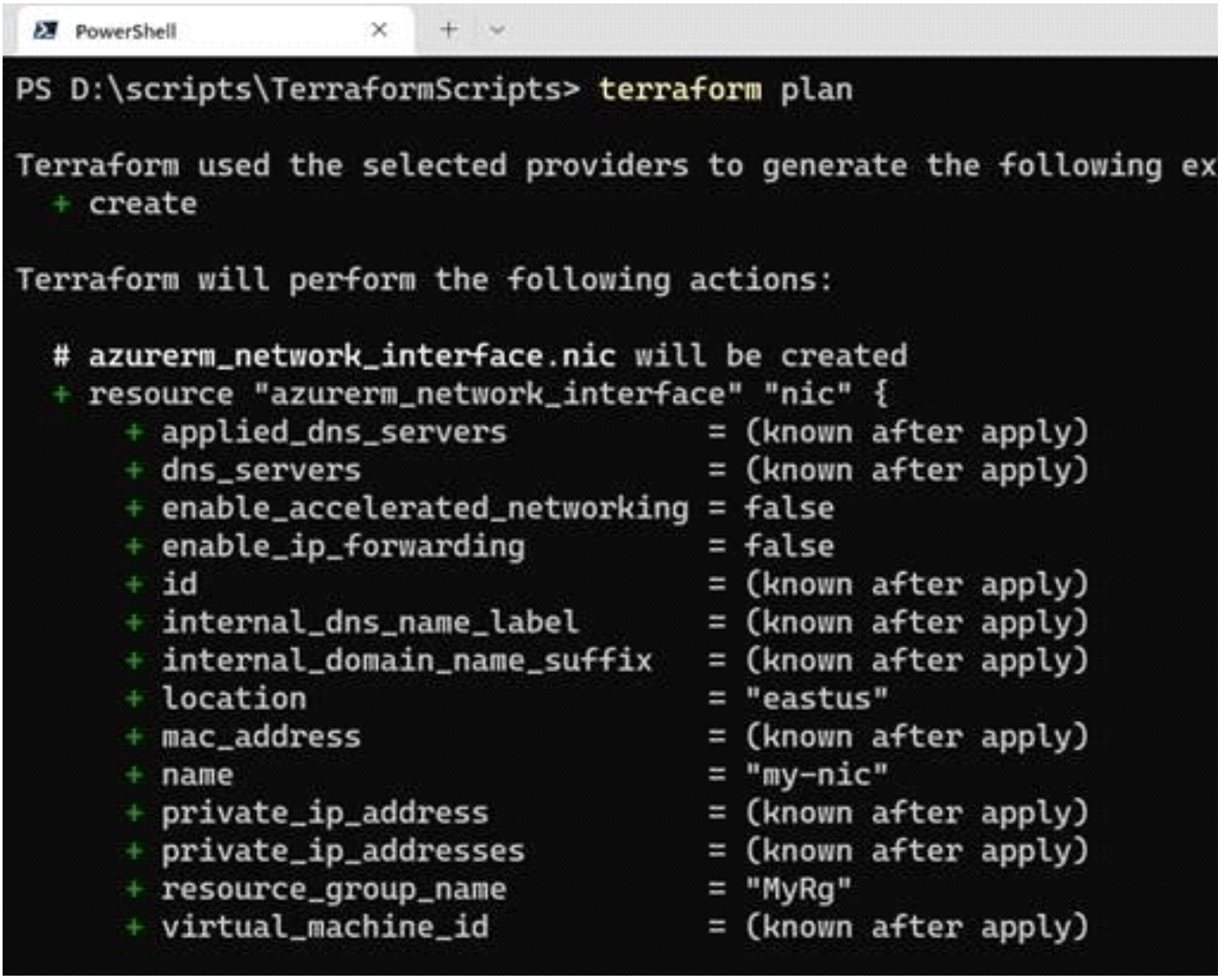
Commands

Friday, May 19, 2023 4:33 PM

#terraform init



#terraform plan


PowerShell
PS D:\scripts\TerraformScripts> terraform plan
Terraform used the selected providers to generate the following ex
+ create
Terraform will perform the following actions:
# azurerm_network_interface.nic will be created
+ resource "azurerm_network_interface" "nic" {
+ applied_dns_servers = (known after apply)
+ dns_servers = (known after apply)
+ enable_accelerated_networking = false
+ enable_ip_forwarding = false
+ id = (known after apply)
+ internal_dns_name_label = (known after apply)
+ internal_domain_name_suffix = (known after apply)
+ location = "eastus"
+ mac_address = (known after apply)
+ name = "my-nic"
+ private_ip_address = (known after apply)
+ private_ip_addresses = (known after apply)
+ resource_group_name = "MyRg"
+ virtual_machine_id = (known after apply)

...

Files required

Friday, May 19, 2023 4:34 PM

Main.tf	Contains code for creating Public facing NIC, Public IP address, VM.
Network.tf	Contains code for virtual network & subnet
Provide.tf	Provider details azure.
Rg.tf	Details for resource group.

Main.tf

Friday, May 19, 2023 4:34 PM

```
#creating public facing NIC
resource "azurerm_network_interface" "nic" {
 name = "my-nic"
 location = "East US"
 resource_group_name = azurerm_resource_group.rg.name
 ip_configuration {
 name ="myipconfig"
 subnet_id = azurerm_subnet.subnet.id
 private_ip_address_allocation = "Dynamic"
 public_ip_address_id = azurerm_public_ip.pip.id
}
}
#creating public IP address
resource "azurerm_public_ip" "pip" {
 name = "mypubipaddress"
 location = "East US"
 resource_group_name = "${azurerm_resource_group.rg.name}"
 allocation_method = "Dynamic"
 domain_name_label = "mydevops"
}

#creating an Azure VM
resource "azurerm_virtual_machine" "vm" {
 name = "myvm"
 location = "East US"
 resource_group_name = azurerm_resource_group.rg.name
 vm_size = "Standard_DS1_v2"
 network_interface_ids = ["${azurerm_network_interface.nic.i
d}"]
 storage_image_reference {
 publisher = "Canonical"
 offer = "UbuntuServer"
 sku = "18.04-LTS"
 version = "Latest"
}
 storage_os_disk {
 name= "myosdisk1"
 caching = "ReadWrite"
 create_option = "FromImage"
 managed_disk_type = "Standard_LRS"
}
 os_profile {
 computer_name= "myvm"
```

```
 admin_username = "jeetu"
 admin_password = "Pa$$w0rd12345"
}
 os_profile_linux_config {
 disable_password_authentication = false
}
}
```

Network.tf

Friday, May 19, 2023 4:34 PM

```
#creating azure vNet
resource "azurerm_virtual_network" "vnet" {
 name = "my-vnet"
 location = "East US"
 address_space = ["172.16.0.0/16"]
 resource_group_name = azurerm_resource_group.rg.name
}
#creating sunet for vNet
resource "azurerm_subnet" "subnet" {
 name = "my-subnet"
 virtual_network_name = azurerm_virtual_network.vnet.name
 resource_group_name = azurerm_resource_group.rg.name
 address_prefixes = ["172.16.1.0/24"]
}
```

Provide.tf

Friday, May 19, 2023 4:35 PM

```
terraform {
 required_providers {
 azurerm = {
 source = "hashicorp/azurerm"
 version = "=3.0.0"
}
}
}
# Configure the Microsoft Azure Provider
provider "azurerm" {
 features {}
}
```

RG.tf

Friday, May 19, 2023 4:35 PM

```
#creating azure RG
resource "azurerm_resource_group" "rg" {
 name = "MyRg"
 location = "East US"
 tags = {
 environment = "Terraform testing"
}
}
```

Creating AWS S3 bucket and adding index.html file

Saturday, November 25, 2023 9:43 PM

Index.html file

```
<html>
<body>
<h1 align='center'>Welcome from Mr. Jeetu</h1>
<marquee>this file is copied from S3 bucket.
</marquee>
</body>
</html>
```

Main.tf file

```
# Defining bucket name - tested
resource "aws_s3_bucket" "mybuck" {
  bucket = "demobeprac0101"
}
# Defining bucket ownership rules - tested
resource "aws_s3_bucket_ownership_controls" "mybuck" {
  bucket = aws_s3_bucket.mybuck.id
  rule {
    object_ownership = "BucketOwnerPreferred"
  }
}
# Allowing bucket to be accessed publicly - tested
resource "aws_s3_bucket_public_access_block" "mybuck"
{
  bucket = aws_s3_bucket.mybuck.id
  block_public_acls       = false
  block_public_policy     = false
  ignore_public_acls      = false
  restrict_public_buckets = false
}
# Defining public read access to the bucket.  - tested
resource "aws_s3_bucket_acl" "mybuck" {
  depends_on = [
    aws_s3_bucket_ownership_controls.mybuck,
    aws_s3_bucket_public_access_block.mybuck,
```

```
  ]
  bucket = aws_s3_bucket.mybuck.id
  acl    = "public-read"
}
# uploading file to S3 bucket - tested
resource "aws_s3_bucket_object" "my_object" {
  bucket = aws_s3_bucket.mybuck.id
  key    = "index.html"
  source = "index.html"
  acl    = "private"
}
```

Provider.tf file

```
# defining the provider
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.26.0"
    }
  }
}
#defining the region to be used.
provider "aws" {
  # Configuration options
  region = "us-east-2"
}
```

Creating VPC, EC2 instance

Saturday, November 25, 2023 9:43 PM

Main.tf file

```
resource "aws_vpc" "new_vpc" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"
  tags = {
    Name = "new_vpc_3"
  }
}
resource "aws_subnet" "sub1" {
  vpc_id     = aws_vpc.new_vpc.id
  cidr_block = "10.0.1.0/24"
  tags = {
    Name = "new_vpc_subnet_1"
  }
}
resource "aws_subnet" "sub2" {
  vpc_id     = aws_vpc.new_vpc.id
  cidr_block = "10.0.2.0/24"
  tags = {
    Name = "new_vpc_subnet_2"
  }
}
# Creating Internet Gateway & attach it to VPC.
resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.new_vpc.id
  tags = {
    Name = "new_vpc_igw"
  }
}
# new RT
resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.new_vpc.id
}
# attaching IGW with the RT
resource "aws_route" "rto" {
  route_table_id         = aws_route_table.rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.gw.id
```

```
}
# associating RT as MAIN ROUTE TABLE
resource "aws_main_route_table_association" "a" {
  vpc_id         = aws_vpc.new_vpc.id
  route_table_id = aws_route_table.rt.id
}
# RT association with subnet(s)
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.sub1.id
  route_table_id = aws_route_table.rt.id
}
resource "aws_route_table_association" "b" {
  subnet_id      = aws_subnet.sub2.id
  route_table_id = aws_route_table.rt.id
}
# CREATING security group
resource "aws_security_group" "sg" {
  name        = "web-sg"
  description = "This security group allows SSH, HTTP,
and HTTPS protocols"
  vpc_id = aws_vpc.new_vpc.id
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allow SSH from
anywhere (adjust as needed)
  }
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allow HTTP from
anywhere (adjust as needed)
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # Allow HTTPS from
anywhere (adjust as needed)
  }
  egress {
```

```
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
    tags = {
    Name = "web-sg"
  }
}
# generating key-pair on EC2 dashboard
resource "aws_key_pair" "web-kp" {
  key_name   = "web-kp"
  public_key = file("C:\\Users\\admin\\OneDrive -
Pioneer Business Solutions\\Desktop\\keypair\\key-
pubic")
}
# # attaching IAM role
#   resource "aws_iam_role_policy" "ec2s3" {
#     name   = "EC2-to-S3-full-access"
#     role   = aws_iam_role.EC2-to-S3-full-access.id
#   }
# Creating EC2 instance
resource "aws_instance" "tfinst" {
  ami           = "ami-06d4b7182ac3480fa"
  instance_type = "t2.micro"
  key_name      = "web-kp"
  subnet_id     = aws_subnet.sub1.id
  vpc_security_group_ids = [aws_security_group.sg.id]
  associate_public_ip_address = true
  iam_instance_profile = "EC2-to-S3-full-access"
  user_data = <<-EOF
    #!/bin/bash
    sudo su -
    sudo yum install -y httpd php && sudo systemctl
start httpd && sudo systemctl enable httpd
    echo "<html>
    <body>
    <h1 align='center'>
    This is a demo
    </h1>
    </body>
    </html> > /var/www/html/index.php
    chmod u+x /var/www/html/index.php
```

```
    EOF
  tags = {
    Name = "tfinstance01"
  }
}
```

Provider.tf file

```
# defining the provider
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.26.0"
    }
  }
}
#defining the region to be used.
provider "aws" {
  region = "us-east-2"
}
```

Introduction to YAML

Wednesday, November 29, 2023 2:10 PM

- It's very important.
- Previous name: Yet another markup language.
- Current name: Yaml Aint Markup Language.

What is markup language?

- It's a system for annotating a document in a way that is syntactically distinguishable from that.

Examples of markup language:

1. Hyper Text Markup Lang. (HTML)
2. Extensible Markup Lang. (XML)
3. Extensible Hypertext Markup Lang (XHTML)
4. Mathematical Markup Lang (MathML)
5. Scalable Vector Graphics (SVG)
6. LaTeX (pronounced as Lay-tech)
7. Markdown
8. Yaml Aint Markup Lang. (YAML)
9. JSON

- YAML is a data format, use to exchange data.
- Similar to XML & JSON
- YAML is not a programming lang.
- It stores some data in a reachable format.
- YAML can store data, not "COMMAND".

Data serialization: convert regular data objects into complex data types.
Data de-serialization: converting of complex data into regular data.

Benefits:

- Simple, easy to read.
- Strict syntax, like indent is important.
- Easily convertible to JSON, XML.
- More powerful, when representing complex data.

Extensions:

1. .YAML
2. .YML

Creating YAML file

YAML code start from "---"

First Code:

```
"apple": "a red fruit"
1: "ways no. 1"
```

```
---
"apple": "a red fruit"
1: "ways no. 1"
```

To create list in YAML:

Code:

```
# to create list in YAML
# list          # -> a comment
- apple
- banana
- mango
- Apple
```

Snip:

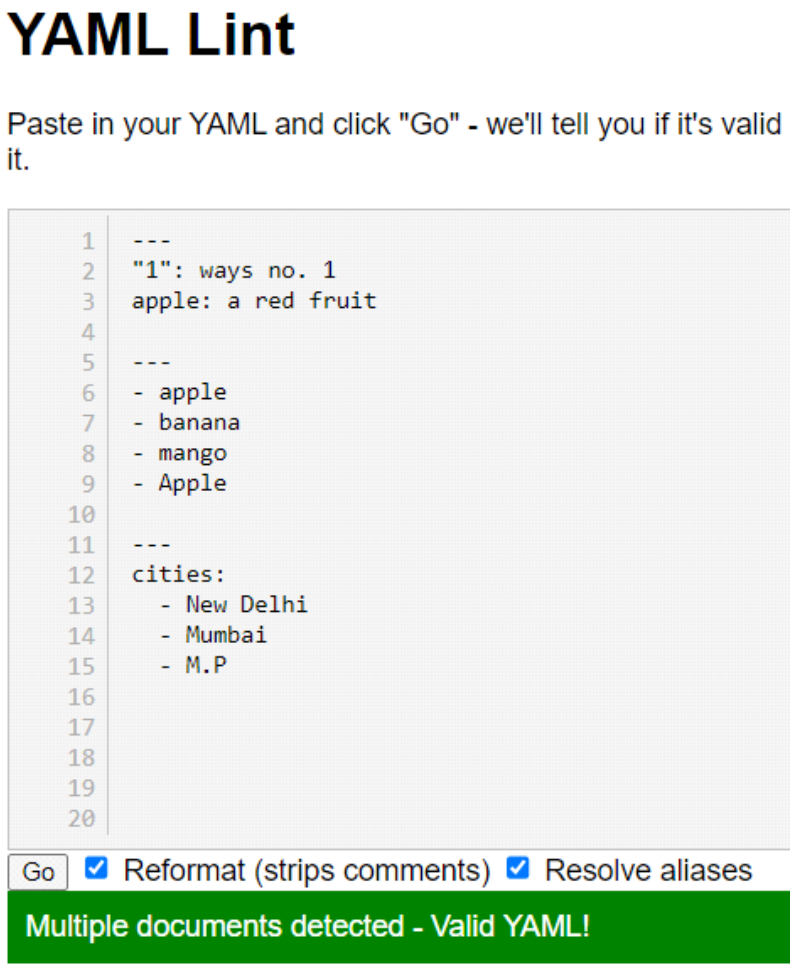
```
# to create list in YAML
# list          # -> a comment
---
- apple
- banana
- mango
- Apple
```

Adding data in block styles:

```yaml
# Adding data in block style:
---
cities:
  - New Delhi
  - Mumbai
  - M.P
```

NOTE: Indentation is important in YAML.

https://www.yamllint.com/



To end a YAML document:

...

To convert YAML to JSON:

https://onlineyamltools.com/

To store data in yaml without indentation:

```yaml
new_cities: [New Delhi, Mumbai, M.P]
---
```

To store data in key-values pairs:

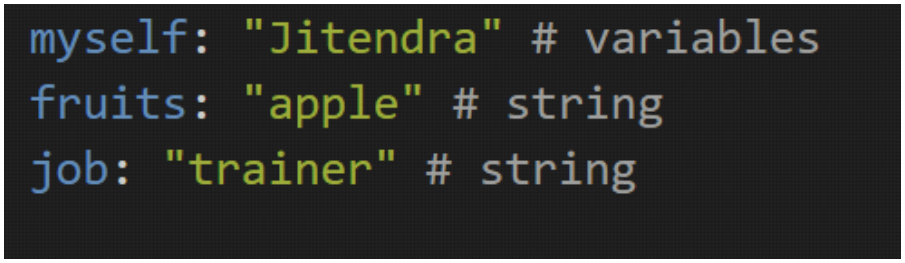
```
# To store data in key-values pairs:
{ mango: "Yellow fruit", age: 30 }
---
```

NOTE: YAML doesn't support multi-line comments.

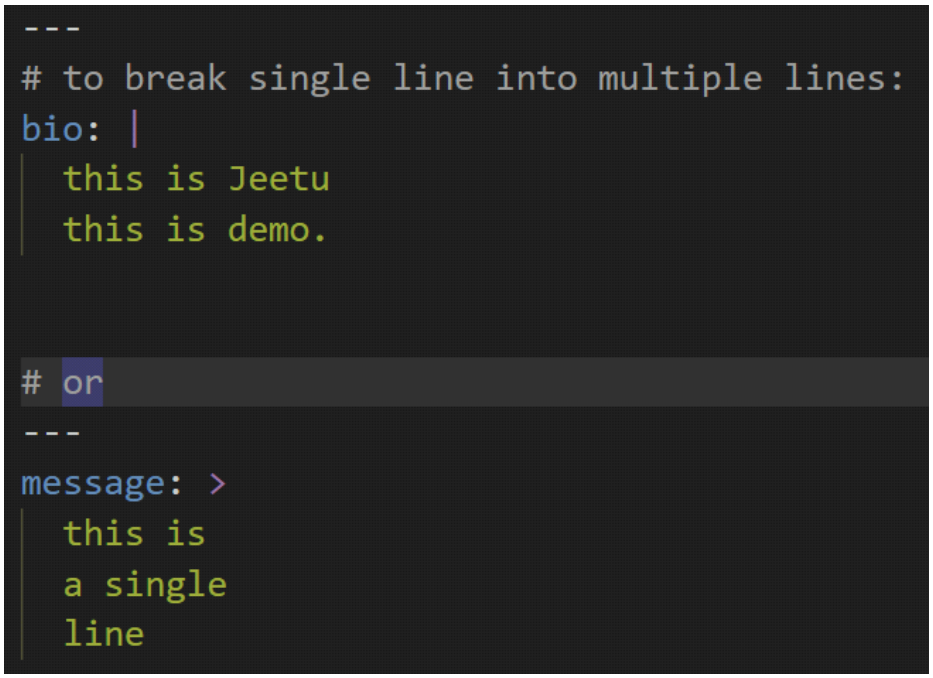
Datatypes in YAML

Wednesday, November 29, 2023 4:03 PM

```yaml
myself: "Jitendra" # variables
fruits: "apple" # string
job: "trainer" # string
---
```




```yaml
# to break single line into multiple lines:
bio: |
  this is Jeetu
  this is demo.

# or
---
message: >
  this is
  a single
  line
```



Creating integer, float and Boolean datatypes:

```yaml
# Integer
number: 12345

# Float
masks: 79.85

# boolean
booleanValue1: NO
booleanValue2: n
booleanValue3: false
booleanValue4: FALSE
booleanValue5: Yes
booleanValue6: Y
booleanValue7: YES
```

Data types:

```yaml
# specifying data types:
zero: !!int 0

# binary number
binaryNum: !!binary 0b1101

# Octal Number
octalNum: !!int 06574

# hexadecimal number
hexa: !!int 0x45

# Comma values
commaValue: !!int +540000
```

Data types:

```yaml
# Floating numbers
marks: !!float 56.35
infitite: !!float .inf
not a no: !!float .nan

# String
my_message: !!str this is a messages

# null
surname: !!null Null #OR
~: this is a null key

# date & time
date: 2023-11-29
dateNTime: 2023-11-29T11:40
ndate: !!timestamp 2023-11-29
ndate2: !!timestamp 2023-11-29T11:41:00 +05:30

# Exponential
expoNos: 6.023E56
```

Advance datatypes

Thursday, November 30, 2023 12:01 PM

```yaml
# Sequence
students: !!seq
  - marks
  - name
  - roll_no
# alternate for seq datatype:
stu: [new_marks, new_name, new_roll_no]
# Empty sequence
sparse seq:
  - hey
  - man
  -
  - Null
  - sup
# Nested sequence
---
- - mango
  - banana
- - mark
  - date
# key-value pairs are called "MAPS"
---
maps:
  - key1: value1
  - key2: value2
  - key3: value3


# Nested mapping: map within map
# method 1
---
name: "Jitendra"
role_no:
  age: 30
  job: "trainer"
...
# method 2
---
name: "Jeetu"
```

```yaml
role: { age: 30, job: "trainer" }
# Pairs - keys may have "duplicate values"
# this will be an array of hashtables
---
pair_example: !!pairs
  - job: studnent
  - job: teacher
#or
pair_ex2: !!pairs [job: student, job: trainer]
# Set - will allow you to have a "unique values"
---
name: !!set
  ? Jeetu
  ? Singh
  ? Tomar
# Dictionary (!!omap)
---
countries:
  Afghanistan: Kabul
  Albania: Tirana
  Algeria: Algiers
  Andorra: Andorra la Vella
  Angola: Luanda


# Reusing properties
# Method 1
liking:
  fav_fruit: mango
  dislikes: grapes
per1:
  name: "Jeetu"
  fav_fruit: mango
  dislikes: grapes
per2:
  name: "Jeetu"
  fav_fruit: mango
  dislikes: grapes
# here, if number of people increased,
# then repeatative-ness will increase.


# Reusing properties using "ANCHORS"
liking1: &likes
```

```yaml
  fav_fruit: mango
  dislikes: grapes
pers1:
  names: "Jeetu"
  <<: *likes
pers2:
  names: "t2"
  <<: *likes
```

Validating YAML files:

Thursday, November 30, 2023 1:02 PM

YAML Lint	https://www.yamllint.com/
Datree	https://www.datree.io/
Monokle	https://monokle.io/
Lens	https://k8slens.dev/

YAML code files

Thursday, November 30, 2023 1:16 PM

datatypes

adv_datatypes

first

YAML basics - Ansible documentation

Friday, March 15, 2024 1:12 AM

- Early every YAML file start with a "list".
- Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”.
- All YAML files (regardless of their association with Ansible or not) can optionally begin with --- and end with --- This is part of the YAML format and indicates the start and end of a document.

Example 1:

```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango
---
```

Example 2:

```
# An employee record
martin:
  name: Martin Devloper
  job: Developer
  skill: Elite
```

Link:
https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html#yaml-syntax

Installing Jenkins using Docker containers on linux

Run the following into the Dockerfile on linux machine:

```
FROM jenkins/jenkins:2.414.2-jdk17
USER root
RUN apt-get update && apt-get install -y lsb-release python3-pip
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.25.3 docker-workflow:1.28"
```

Build the image from this:

```
docker build -t myjenkins-app-1 .
```

Create docker network:

```
docker network create jenkins
```

Creating docker container:

```
docker run --name jenkins-app --restart=on-failure --detach \
  --network jenkins --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 \
  --publish 8080:8080 --publish 50000:50000 \
  --volume jenkins-data:/var/jenkins_home \
  --volume jenkins-docker-certs:/certs/client:ro \
  myjenkins-app-1
```

Check the docker container:

```
docker ps
```

login to web browser:

```
127.0.0.1:8080
```

Execute the below cmd to list PWD for jenkins

```
docker exec jenkins-app cat /var/jenkins_home/secrets/initialAdminPassword
```

--- copy the password & paste it into the web browser.

While installation:

```
# install suggested plugins.

# give
- username
```

- password
- name
- save & continue.

DEMO - 1:

On Web browser

- Browse through the UI.
- Create a new/first job
- Without filling GitHub URL.
- In BUILD, select "*execute shell*" & write some commands like:
  - echo "Hello world"
    - Save & build project
    - Check the console logs
- Then edit the shell again with environmental variables
  - echo "Build ID is: ${BUILD_ID}"
  - echo "Build URL is: ${BUILD_URL}"
    - Save & build project
    - Check the console logs
- Then edit the shell again:
  - Create file and verify it on browser & container location as well.
    - echo "this is Jenkins Pipeline" > ReadMe.txt
    - Save & build project
    - Check the console logs

Now, login to docker container

To access the file system in jenkins container within docker

```
docker exec -it jenkins-app bash
```

Execute commands:

pwd
ls -lrt
cat ReadMe.txt

 # go to

/var/jenkins_home/workspace/my-first-proj-1

DEMO - 2 :- Fetching info from GitHub account

- Create a new job.
- Provide the GitHub URL in the source code management under 'repository URL'.
  https://github.com/jitendrastomar5593/gitdemo
- In Build, select 'execute shell' & run below command.
  Python3 helloworld.py
  - Save & check.
  - Build it.
  - Check console.

==========

Dashboard --> Manage Jenkins

--> Plugins
- Install blueOcean plugins.

--> Manage Nodes & Clouds --> cloud providers
- Docker
- Amazon EC2 (time taking process)
    - Check Install after restart.

Integrating GitHub with Jenkins on AWS EC2 (Ubuntu)

Saturday, March 16, 2024 11:16 PM

1. Create an Ubuntu VM on any cloud (AWS) with port number
   a. 80
   b. 443
   c. 8080 (for Jenkins application)
   d. 22
2. Connect to the EC2 instance using SSH via
   a. Putty
   b. MT Putty
   c. MobaXterm
   d. PowerShell
   e. Windows Terminal
3. Update the ubuntu server and install docker on it.
   a. Set up Docker's apt repository

```
# Add Docker's official GPG key:
sudo apt-get update -y
sudo apt-get install ca-certificates curl -y
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update -y
```

   b. Install docker packages

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

   c. Verify that the Docker Engine installation is successfully

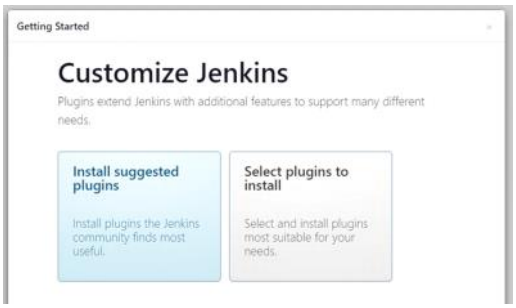
```
docker run hello-world
```

      # systemctl enable docker
      # systemctl start docker
      # usermod -a -G docker <ubuntu>

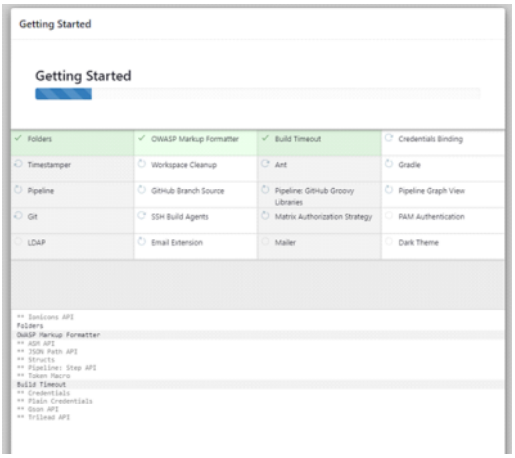
   d. Pulling Jenkins image from dockerhub.
      # docker pull jenkins/jenkins
   e. Listing images:
      # docker images
   f. Create a new directory:
      # mkdir Jenkins
   g. Running Jenkins container on port 8080
      # docker run -d --name jenkins -p 8080:8080 -v $PWD/jenkins/ jenkins/jenkins
      # docker ps
   h. Login to <EC2-INSTANCE-PUBLIC-IP>:<PORT-NUMBER>
      http://<ec-instance>:<8080>
      Unlock the Jenkins using
      # docker ps
      # docker exec -it <e580724f6f66> cat /var/jenkins_home/secrets/initialAdminPassword

Copy this password and paste it to browser page.

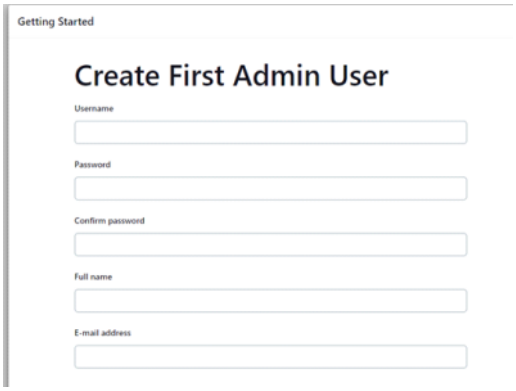
Select **install suggested plugins**:

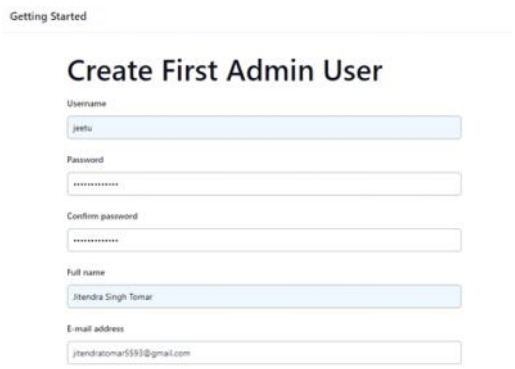


Wait for it:



Fill the below details:

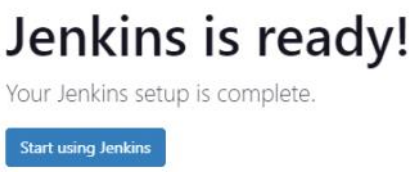




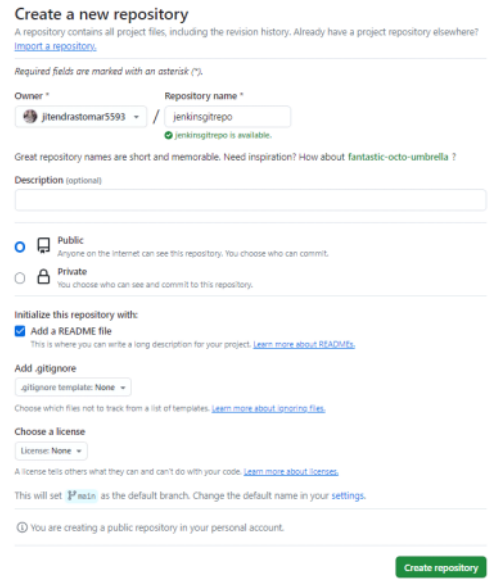
Save & continue.

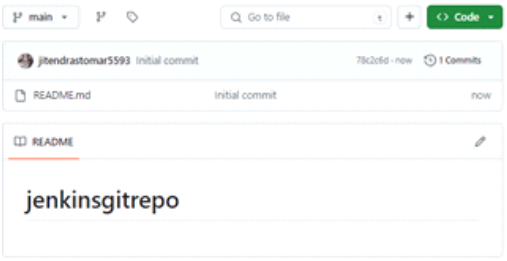
Check the URL: http://<IP-OF-EC2-INSTANCE>:8080/

Save & finish.

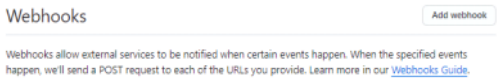


Switch to GitHub & create a new public repository

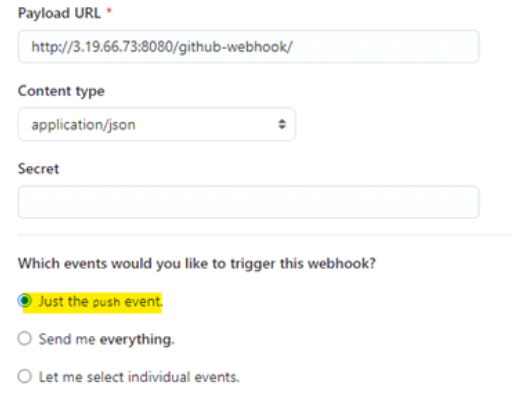




On GitHub, go to “settings” & then “webhooks”. Click “Add webhook”.



Fill details:

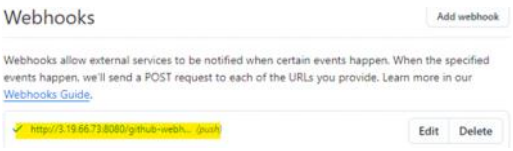


Payload URL → http://<jenkins-svr-public-IP>:<port>/github-webhook/

Content type → application/json

Which events would you like to trigger this webhook → Just the push event

Ensure “Active” is check marked. & click on “Add webhook”. & wait for some time.

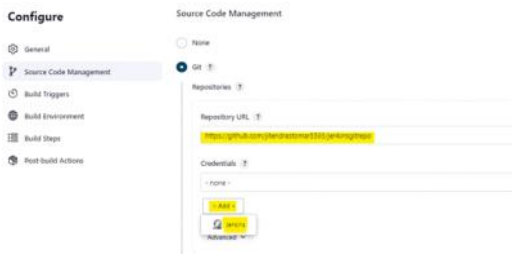


Now switch back Jenkins portal, Jenkins dashboard: - Create “new item”.

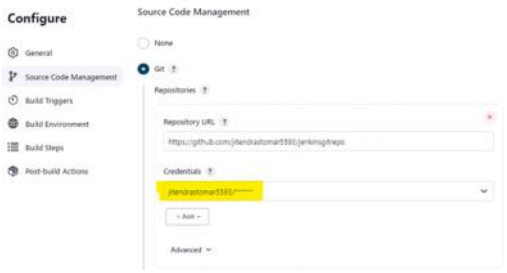
Enter an item name:

Click OK.

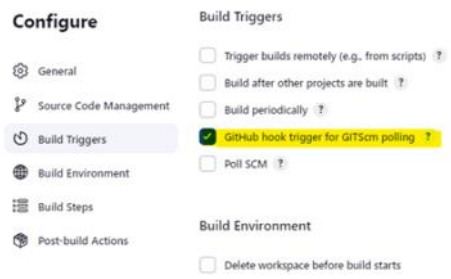
Under “Configure” select “source code management”, select GIT & fill ‘repository URL’.



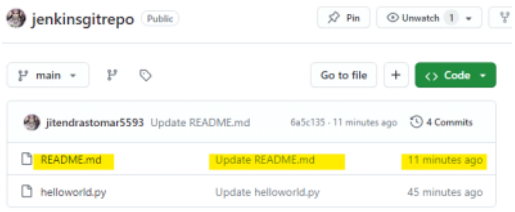
Select “Jenkins” & under credentials, select username & password & fill those.



Click on “Build Triggers”. Select GitHub hook.



Click “Build Steps”, Nothing to select and save. Switch back to GitHub & edit the README.md file & commit it.



Go back to jenkins dashboard page & wait for build history to show up, automatically.

Creating another file & commit on GitHub & switch to Jenkins dashboard and visit build history.

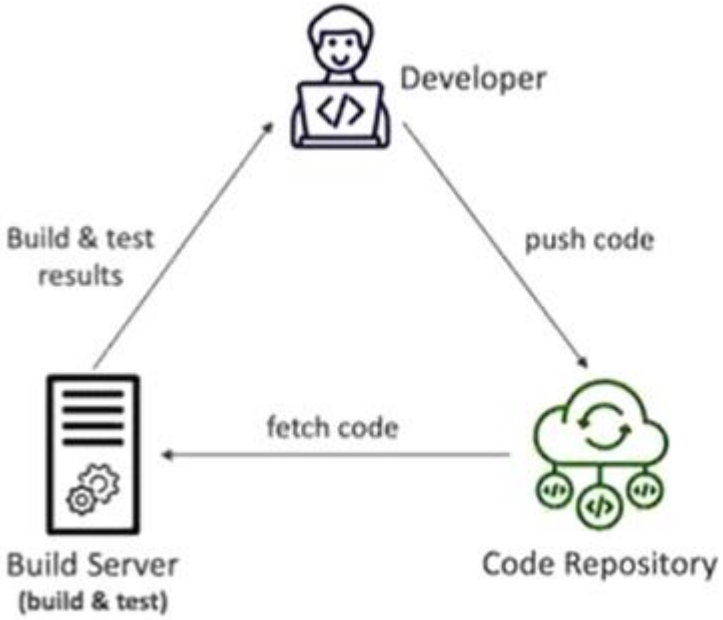
From <https://d.docs.live.net/4b07f4230dd85219/Documents/Jenkins%20documentation%20-%20working%20docx.docx>

Basic Terms

AWS CodePipeline	Automating our pipeline from code to Elastic Beanstalk
AWS CodeBuild	Building & Testing our code
AWS CodeDeploy	Deploying the code to EC2 instances
AWS CodeStar	Manage software development activities in one place.
AWS CodeArtifact	Store, publish & share software packages
AWS CodeGuru	Automated code reviews using machine learning

Continuous Integration (CI)

- Developers pushes the code to a code repository (Ex: GitHub, CodeCommit, Bitbucket).
- A testing / build server checks the code as soon as it's pushed (CodeBuild, Jenkins CI…).
- The developers get feedback about the tests & checks that have passed/failed.

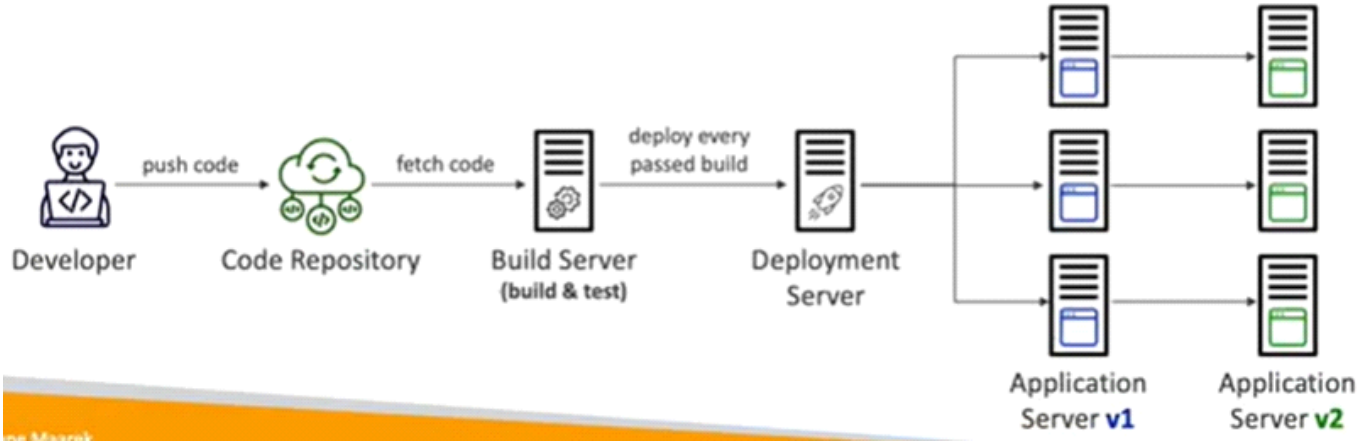


Advantages of this approach:

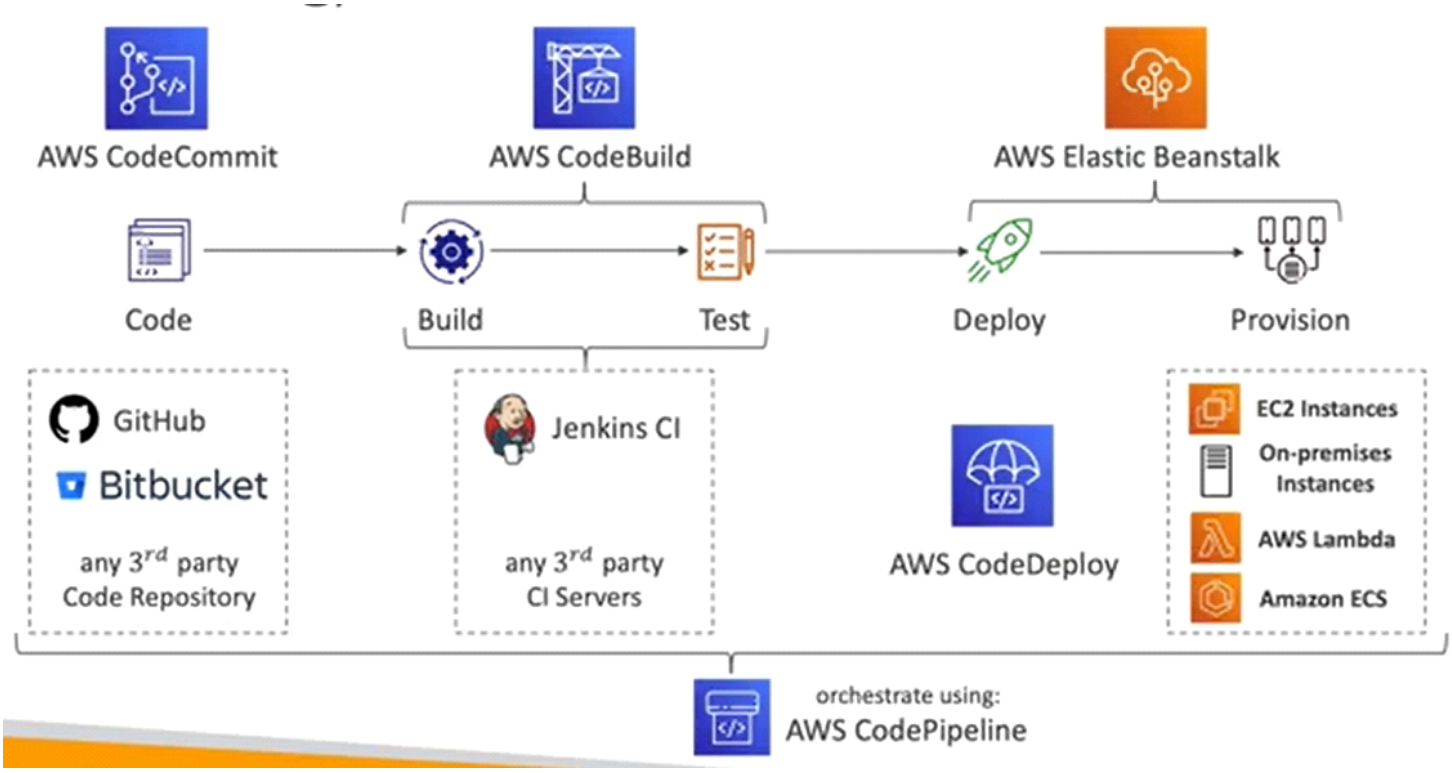
- Find bugs early, then fix bugs.
- Delivers faster as the code is tested.
- Deploy often
- Happier developers, as they're unblocked.

Continuous Delivery (CD)

- Ensures that the software can be release reliably whenever needed.
- Ensures deployments happen often and are quick.
- Shift away from "one release every 3 months" to "5 releases a day".
- Code is deployed automatically. (CodeDeploy, Jenkins CD, Spinnaker).



Technology stack for CI/CD



Stack	3rd party	AWS alternative
Code Repository	GitHub, Bitbucket	AWS CodeCommit
Building & Testing	Jenkins CI	AWS CodeBuild
Deployment	---	AWS Elastic Beanstalk
Provision	---	EC2 instances, On-prem, AWS Lambda, AWS ECS

SQL Query

Friday, April 26, 2024 12:20 AM

 # create database
```
CREATE DATABASE emp;
```

 # select database
```
USE emp;
```

 # creating employee table
```
CREATE TABLE employees (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50),
   age INT,
   position VARCHAR(50),
   salary DECIMAL(10, 2)
);
```

 # adding data into a table
```
INSERT INTO employees (name, age, position, salary)
VALUES ('John Wick', 30, 'Software Engineer', 60000.00);
```

#listing
```
SELECT * FROM employees;
```

 # adding entries into a table
```
INSERT INTO employees (name, age, position, salary) VALUES
('John Doe', 30, 'Manager', 50000.00),
('Jane Smith', 25, 'Developer', 60000.00),
('Michael Johnson', 35, 'Analyst', 55000.00),
('Emily Davis', 28, 'Designer', 48000.00),
('David Wilson', 40, 'Engineer', 70000.00),
('Sarah Brown', 33, 'Marketing', 52000.00),
('Chris Taylor', 29, 'Sales', 53000.00),
('Jennifer Lee', 31, 'HR', 48000.00),
('Kevin Clark', 27, 'QA', 54000.00),
('Amanda Evans', 32, 'Operations', 56000.00);
```

 # select query
```
SELECT *
FROM employees
WHERE name = 'John Doe';
```

 # list user where salary is greater than 50000
```
SELECT *
FROM employees
WHERE salary > 50000;
```

 # list user where salary is greater than 50000 & age is greater than 30

```
SELECT *
FROM employees
WHERE salary > 50000 AND age > 30;

# deleting entries having age more than 40
DELETE FROM emp WHERE age > 40;



# creating table with primary key.
CREATE TABLE emp (
    emp_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    position VARCHAR(50),
    salary DECIMAL(10, 2)
);


# inserting values
```

1. Installing Git

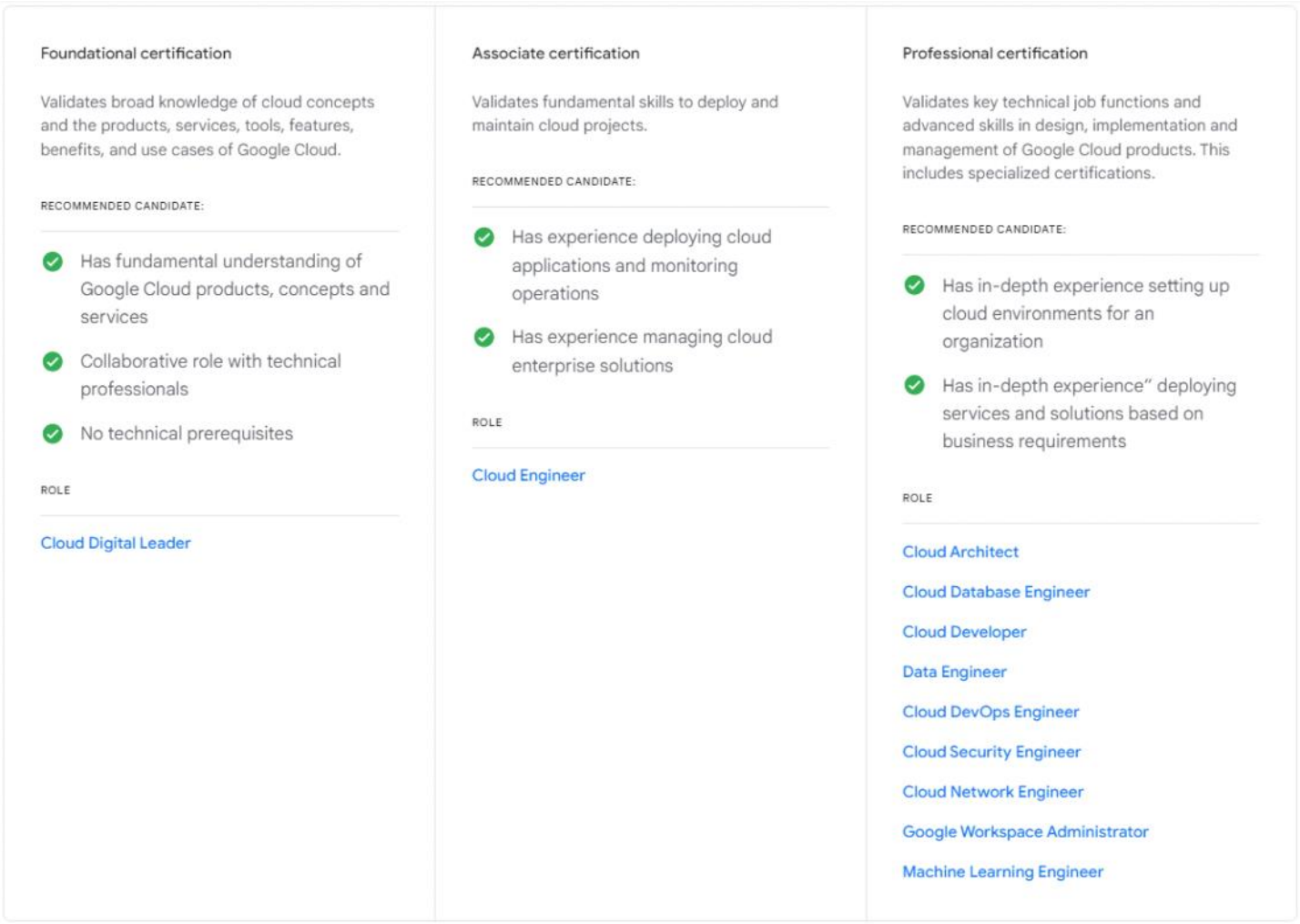
Wednesday, June 12, 2024 7:25 PM

URL: https://www.git-scm.com/download/win

2. GC certifications & Exam

Wednesday, June 12, 2024 7:31 PM

Which Google Cloud certification is right for you?



Cloud Digital Leader: https://cloud.google.com/learn/certification/cloud-digital-leader
Cloud Engineer: https://cloud.google.com/learn/certification/cloud-engineer
Cloud Architect: https://cloud.google.com/learn/certification/cloud-architect
Cloud Database Engineer: https://cloud.google.com/learn/certification/cloud-database-engineer
Cloud Developer: https://cloud.google.com/learn/certification/cloud-developer
Data Engineer: https://cloud.google.com/learn/certification/data-engineer
Cloud DevOps Engineer: https://cloud.google.com/learn/certification/cloud-devops-engineer
Cloud Security Engineer: https://cloud.google.com/learn/certification/cloud-security-engineer
Cloud Network Engineer: https://cloud.google.com/learn/certification/cloud-network-engineer
Google Workspace Administrator: https://cloud.google.com/learn/certification/google-workspace-administrator
Machine Learning Engineer: https://cloud.google.com/learn/certification/machine-learning-engineer

Preferred one is:

Cloud engineer --> **Cloud architect.**

3. Associate Cloud Engineer - TOC

Wednesday, June 12, 2024 7:53 PM

Section 1: Setting up a cloud solution environment (~17.5% of the exam)

1.1 Setting up cloud projects and accounts. Activities include:

- Creating a resource hierarchy
- Applying organizational policies to the resource hierarchy
- Granting members IAM roles within a project
- Managing users and groups in Cloud Identity (manually and automated)
- Enabling APIs within projects
- Provisioning and setting up products in Google Cloud's operations suite

1.2 Managing billing configuration. Activities include:

- Creating one or more billing accounts
- Linking projects to a billing account
- Establishing billing budgets and alerts
- Setting up billing exports

1.3 Installing and configuring the command line interface (CLI), specifically the Cloud SDK (e.g., setting the default project).

Section 2: Planning and configuring a cloud solution (~17.5% of the exam)

2.1 Planning and estimating Google Cloud product use using the Pricing Calculator

2.2 Planning and configuring compute resources. Considerations include:

- Selecting appropriate compute choices for a given workload (e.g., Compute Engine, Google Kubernetes Engine, Cloud Run, Cloud Functions)
- Using preemptible VMs and custom machine types as appropriate

2.3 Planning and configuring data storage options. Considerations include:

- Product choice (e.g., Cloud SQL, BigQuery, Firestore, Spanner, Bigtable)
- Choosing storage options (e.g., Zonal persistent disk, Regional balanced persistent disk, Standard, Nearline, Coldline, Archive)

2.4 Planning and configuring network resources. Tasks include:

- Differentiating load balancing options
- Identifying resource locations in a network for availability
- Configuring Cloud DNS

Section 3: Deploying and implementing a cloud solution (~25% of the exam)

3.1 Deploying and implementing Compute Engine resources. Tasks include:

- Launching a compute instance using the Google Cloud console and Cloud SDK (gcloud) (e.g., assign disks, availability policy, SSH keys)
- Creating an autoscaled managed instance group using an instance template
- Generating/uploading a custom SSH key for instances
- Installing and configuring the Cloud Monitoring and Logging Agent
- Assessing compute quotas and requesting increases

3.2 Deploying and implementing Google Kubernetes Engine resources. Tasks include:

- Installing and configuring the command line interface (CLI) for Kubernetes (kubectl)
- Deploying a Google Kubernetes Engine cluster with different configurations including AutoPilot, regional clusters, private clusters, etc.
- Deploying a containerized application to Google Kubernetes Engine
- Configuring Google Kubernetes Engine monitoring and logging

3.3 Deploying and implementing Cloud Run and Cloud Functions resources. Tasks include, where applicable:

- Deploying an application and updating scaling configuration, versions, and traffic splitting

- Deploying an application that receives Google Cloud events (e.g., Pub/Sub events, Cloud Storage object change notification events)

3.4 Deploying and implementing data solutions. Tasks include:

- Initializing data systems with products (e.g., Cloud SQL, Firestore, BigQuery, Spanner, Pub/Sub, Bigtable, Dataproc, Dataflow, Cloud Storage)

- Loading data (e.g., command line upload, API transfer, import/export, load data from Cloud Storage, streaming data to Pub/Sub)

3.5 Deploying and implementing networking resources. Tasks include:

- Creating a VPC with subnets (e.g., custom-mode VPC, shared VPC)

- Launching a Compute Engine instance with custom network configuration (e.g., internal-only IP address, Google private access, static external and private IP address, network tags)

- Creating ingress and egress firewall rules for a VPC (e.g., IP subnets, network tags, service accounts)

- Creating a VPN between a Google VPC and an external network using Cloud VPN

- Creating a load balancer to distribute application network traffic to an application (e.g., Global HTTP(S) load balancer, Global SSL Proxy load balancer, Global TCP Proxy load balancer, regional network load balancer, regional internal load balancer)

3.6 Deploying a solution using Cloud Marketplace. Tasks include:

- Browsing the Cloud Marketplace catalog and viewing solution details

- Deploying a Cloud Marketplace solution

3.7 Implementing resources via infrastructure as code. Tasks include:

- Building infrastructure via Cloud Foundation Toolkit templates and implementing best practices

- Installing and configuring Config Connector in Google Kubernetes Engine to create, update, delete, and secure resources

Section 4: Ensuring successful operation of a cloud solution (~20% of the exam)

4.1 Managing Compute Engine resources. Tasks include:

- Managing a single VM instance (e.g., start, stop, edit configuration, or delete an instance)
- Remotely connecting to the instance
- Attaching a GPU to a new instance and installing necessary dependencies
- Viewing current running VM inventory (instance IDs, details)
- Working with snapshots (e.g., create a snapshot from a VM, view snapshots, delete a snapshot)
- Working with images (e.g., create an image from a VM or a snapshot, view images, delete an image)
- Working with instance groups (e.g., set autoscaling parameters, assign instance template, create an instance template, remove instance group)
- Working with management interfaces (e.g., Google Cloud console, Cloud Shell, Cloud SDK)

4.2 Managing Google Kubernetes Engine resources. Tasks include:

- Viewing current running cluster inventory (nodes, pods, services)
- Browsing Docker images and viewing their details in the Artifact Registry
- Working with node pools (e.g., add, edit, or remove a node pool)
- Working with pods (e.g., add, edit, or remove pods)
- Working with services (e.g., add, edit, or remove a service)
- Working with stateful applications (e.g. persistent volumes, stateful sets)
- Managing Horizontal and Vertical autoscaling configurations
- Working with management interfaces (e.g., Google Cloud console, Cloud Shell, Cloud SDK, kubectl)

4.3 Managing Cloud Run resources. Tasks include:

- Adjusting application traffic-splitting parameters
- Setting scaling parameters for autoscaling instances
- Determining whether to run Cloud Run (fully managed) or Cloud Run for Anthos

4.4 Managing storage and database solutions. Tasks include:

- Managing and securing objects in and between Cloud Storage buckets
- Setting object life cycle management policies for Cloud Storage buckets
- Executing queries to retrieve data from data instances (e.g., Cloud SQL, BigQuery, Spanner, Datastore, Bigtable)
- Estimating costs of data storage resources
- Backing up and restoring database instances (e.g., Cloud SQL, Datastore)
- Reviewing job status in Dataproc, Dataflow, or BigQuery

4.5 Managing networking resources. Tasks include:

- Adding a subnet to an existing VPC
- Expanding a subnet to have more IP addresses
- Reserving static external or internal IP addresses
- Working with CloudDNS, CloudNAT, Load Balancers and firewall rules

4.6 Monitoring and logging. Tasks include:

- Creating Cloud Monitoring alerts based on resource metrics
- Creating and ingesting Cloud Monitoring custom metrics (e.g., from applications or logs)
- Configuring log sinks to export logs to external systems (e.g., on-premises or BigQuery)
- Configuring log routers
- Viewing and filtering logs in Cloud Logging
- Viewing specific log message details in Cloud Logging

- Using cloud diagnostics to research an application issue (e.g., viewing Cloud Trace data, using Cloud Debug to view an application point-in-time)
- Viewing Google Cloud status

Section 5: Configuring access and security (~20% of the exam)

5.1 Managing Identity and Access Management (IAM). Tasks include:

- Viewing IAM policies
- Creating IAM policies
- Managing the various role types and defining custom IAM roles (e.g., primitive, predefined and custom)

5.2 Managing service accounts. Tasks include:

- Creating service accounts
- Using service accounts in IAM policies with minimum permissions
- Assigning service accounts to resources
- Managing IAM of a service account
- Managing service account impersonation
- Creating and managing short-lived service account credentials

5.3 Viewing audit logs

From <https://cloud.google.com/learn/certification/guides/cloud-engineer>

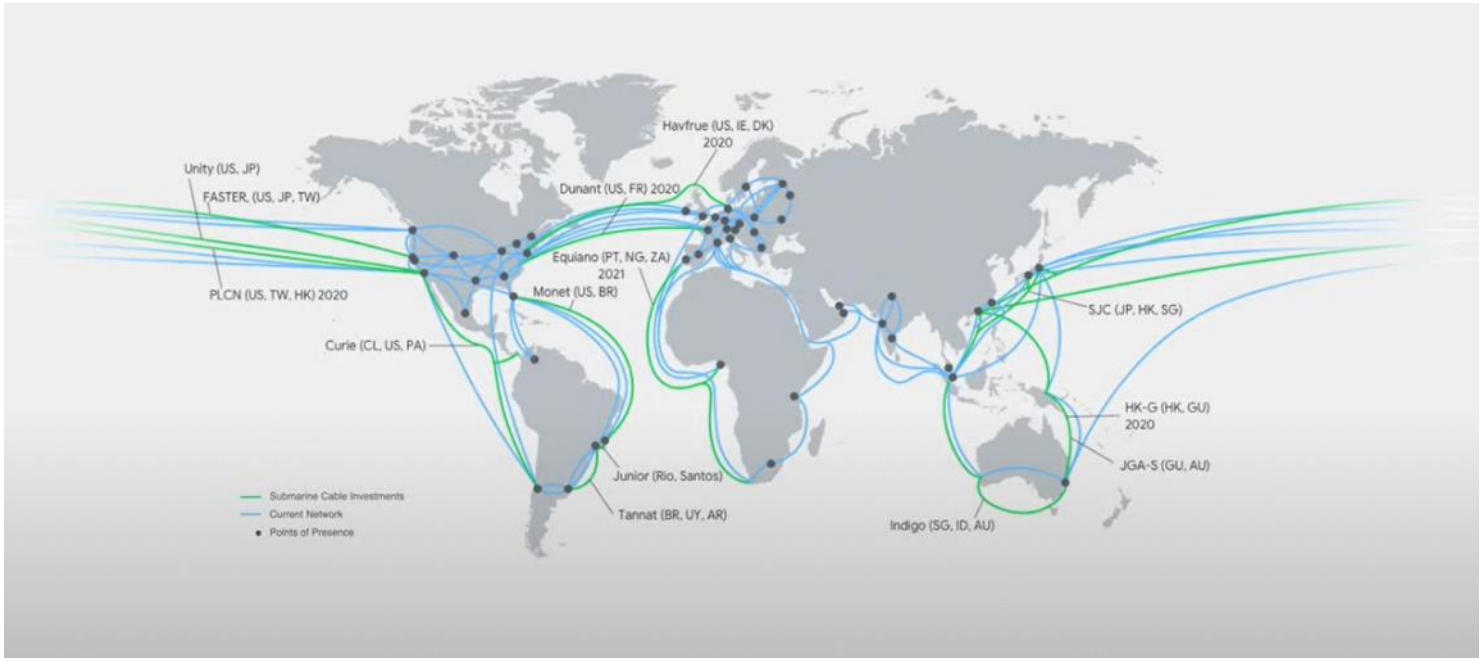
4. Cloud Fundamentals

Thursday, June 13, 2024 12:38 AM

1. What is cloud computing?
2. Cloud Deployment models
    a. Public cloud
    b. Private cloud
    c. Hybrid cloud
3. Cloud service models
    a. IAAS
    b. PAAS
    c. SAAS

5. GCP Global Infrastructure

Thursday, June 13, 2024 12:40 AM



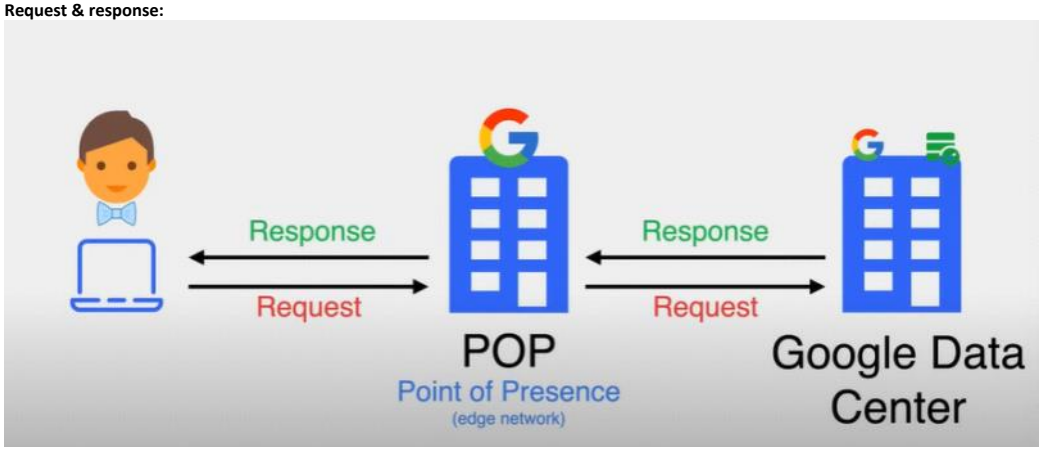
Currently

- 13 subsea cables
  - Subsea cable connects 2 or more continents together.
- Hundreds of thousands of miles of fiber cable

Global footprint

- 24 regions
- 73 zones
- 114 Network edge locations
- 200+ countries & territories

Request & response:



Zones

- A deployment area within a region
- It's the smallest entity.

Region

- Geographical area.
- Divided into sub-zones
- For FT & HA, intercommunication within zones is <5ms (less than 5 milliseconds) within a region.

Multi-Region

- Large geo areas that contains 2 or more regions.
- Allows Google services to maximize redundancy & distribution within & across regions.
- This is basically for high availability.

6. Compute Service Options

Thursday, June 13, 2024 12:50 AM

apc