08 - 30 Jan 2023 (Ansible)

25 January 2023

18:42

Ansible

* 1. Create one ansible server
  2. It is agent less model. It is Push model.
  3. To allow ansible server to access other servers it should have
     1. Ip address
     2. Username
     3. Password / key
  4. IP address passed in the form of inventory file, it contains IP addresses.
  5. It needs Python as a prerequisite. Ansible is create din Python
  6. It is an open source software.
  7. It also has enterprise version which requires license, this one has UI and user management features.

# Install Ansible

* 1. <https://github.com/lerndevops/labs/tree/master/ansible/install>

sudo apt-get update

sudo apt-get install software-properties-common

sudo apt-add-repository --yes --update ppa:ansible/ansible

sudo apt-get install ansible

* 1. Ansible --version

# How to play Ansible on remote servers

Ansible has pre-defined ansible modules.

* 1. Ad-hoc way / ad-hoc changes
  2. Scripted way using play books

## How to write playbook . Script?

YAML/YML

# Create Inventory File

* 1. Ansible default configuration file: **/ect/ansible/ansible.cfg <it is not recommended>**
  2. Don't tuooch default files. Take a copy of ansible.cfg and hosts files to
  3. Cd /home/ubunu
  4. cp /etc/ansible/ansible.cfg .
  5. cp /etc/ansible/hosts .
  6. Cat ansible.cfg
     1. It will show some error, copy the command and execute to read it
  7. In **hosts** file, delete all existing values and add 2 IP addresses
  8. Server requries username which is ubuntu and password is pem key
  9. Pem key is in local laptop, copy
  10. From local laptop, copy command
      1. **scp -i edujan16.pem <file-to-copy> ubuntu@35.78.106.76:/home/ubuntu**scp -i edujan16.pem edujan16.pem ubuntu@18.181.78.41:/home/ubuntu

* 1. Test connection
     1. Ssh -i edujan16.pem ubuntu@vmIP

## 

## Ad-hoc way / ad-hoc changes

* 1. Test one ansible command
     1. ansible --inventory /home/ubuntu/hosts all -m **ping -u** ubuntu --private-key edujan16.pem
     2. This time it asked for yes or no, post key verification failed
     3. To avoid this us econfig file
  2. In config file ansible.cfg, remove semi colon (;) and set this value
     1. Host\_key\_checking = False
  3. Disk utilization o linux machine:-ddf -h. test some remote commnds
     1. Ansible --inventory /home/ubuntu/hosts all -m **shell -a "dh -h"** -u ubuntu --private-key edujan16.pem
     2. Ansible --inventory /home/ubuntu/hosts all -m **shell -a "free -mh"** -u ubuntu --private-key edujan16.pem
     3. Ansible --inventory /home/ubuntu/hosts all -m **shell -a "hostname"** -u ubuntu --private-key edujan16.pem
  4. To copy
     1. Ansible --inventory /home/ubuntu/hosts all -m **copy -a "src=/home/ubuntu/ansble.cfg dest=/temp"** -u ubuntu --private-key edujan16.pem
     2. Ansible --inventory /home/ubuntu/hosts all -m **shell -a "ls -l /tmp"** -u ubuntu --private-key edujan16.pem
  5. If a change already implemented then it will skip it if executed next time.
  6. Every ad-hoc command is individual change
  7. **How many modules are there?**
     1. It has many modules.
     2. ansible-doc -l
     3. There are over 3000 modules
     4. This is how we search for a module: ansible-doc -l | grep copy
     5. Info about copy module:  
        ansible-doc copy
     6. <https://docs.ansible.com> contains complete information about modules

# Host group

* 1. We can have multiple inventory file
  2. myinv file  
     [dev]  
     IP1  
     IP2  
       
     [qa]  
     ip1  
     ip2  
     10.10.10.1[0:9]  
       
     [prod]  
     dnsname1  
     dnsname2

[webservers]  
[linuxservers]  
[winowsserver]  
[dbservers]  
…..

* 1. Cat myinv  
     Ansible --inventory /home/ubuntu/hosts **prod** -m **shell -a "ls -l /tmp"** -u ubuntu --private-key edujan16.pem  
     Ansible --inventory /home/ubuntu/hosts **dev,qa**-m **shell -a "ls -l /tmp"** -u ubuntu --private-key edujan16.pem

## What if username and password is different for all servers

* 1. In host file  
     [dev]  
     10.10.10.10 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/home/ubuntu/edujan16.pem  
     10.10.10.11 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/home/ubuntu/edujan16.pem  
       
     [qa]  
     10.10.10.12 ansible\_user=devops ansible\_ssh\_pass=today@1234  
     10.10.10.13 ansible\_user=devops ansible\_ssh\_pass=today@1234
  2. **What if there are many servers?**

[dev]  
10.10.10.10 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/home/ubuntu/edujan16.pem  
10.10.10.11 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/home/ubuntu/edujan16.pem  
  
  
[qa]  
10.10.10.12  
10.10.10.13  
  
[qa:vars]  
ansible\_user=devops  
ansible\_ssh\_pass=today@1234  
  
[dev:vars]  
ansible\_user=ubuntu  
ansible\_ssh\_private\_key\_file=/home/ubuntu/edujan16.pem

* 1. Execute this way. No need to pass username and passsword/key  
     Ansible --inventory /home/ubuntu/hosts all -m **shell -a "ls -l /tmp"**

<https://drive.google.com/drive/folders/10j0iSusze8HPxOd9Ni0mbGE5UQWapZFn?usp=sharing>

## Scripted way using playbooks

In tomorrow's

09 - 31 Jan 2023 (YAML,Ansible)

31 January 2023

20:35

# Ansible - Scripted Way

# YAML

YAML / YML - data serialization language

Based on indentation

DO NOT USE TAB KEY

Yaml is space sensitive / yaml is case-sensitive

All the data in yaml we write as a key value pairs

**key: value**

Human friendly lang.

Yaml config file is used in every devops tool like ansible, kubernetes, etc.

# 3 data structures of YAML

1. scalar  
   abc.yaml  
   name: naresh #name=naresh  
   a: 6  
   bbc: 5.6
2. dict/map  
   #describe an object  
   laptop:  
    ram: "16 gb"  
    cpu: 8core  
    make: apple  
    os: macos
3. list/array  
   basket.yaml  
   - choc  
   - soap  
   - coke

## Example

Uber:

- name: audi  
 color: red  
 model: sedan

- bmw  
 color:

- white

- red  
 model:

- xuv

- sedan  
 engine:

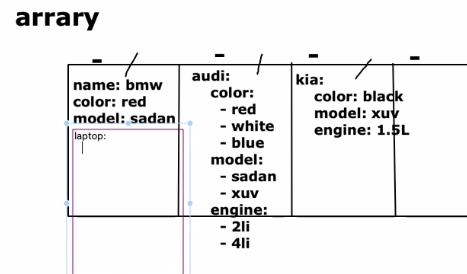
- 2lr

- 4lr

- kia

color: blue  
 model: compact-xuv

engine: 1.5l



# Playbook

Keys are mostly pre-defined by technology like ansible.

You may change the value of key as per requirement.

You may use predefined key and value.

Ansible Terminology

TASK

-- a single change that we implement like install java, check disk status, etc.

-- use pre-defined module

-- single change task = single pre-defined module of ansible

Ex: copy / shell / ping

PLAY

-- list of tasks, multiple changes, that we want to perform on list or group of all remote servers.

-- list of tasks = list of ansible pre-defined modules

-- we can get list of servers from inventory file that we prepared

-- multiple changes on multiple servers

PLAYBOOK

-- is a script file written in yaml

-- playbook can consist of a either single play OR list of plays

Playbook.yaml  
- name: PLAY1  
 hosts: prod #this is the group from inventory file  
 tasks:  
 - name: "task1 - create file"  
 file:  
 path: /tmp/hello.txt  
 state: touch  
 - name: "task2 - copy a file"  
 copy:  
 src: /etc/hosts #the source is always the controller machine here  
 dest: /tmp  
  
- name: PLAY2  
 hosts: qa #this is the group from inventory file  
 tasks:  
 - name: "task1 - create file"  
 file:  
 path: /tmp/qa.txt  
 state: touch  
 - name: "task2 - copy a file"  
 copy:  
 src: /etc/hosts/resolv.config #the source is always the controller machine here  
 dest: /tmp

Copy paste in linux server

Check yaml file, is there any indentation error or not

Ansible-playbook --inventory myinv playbook.yaml --syntax-check

If file name appeared the all ok.

This will execute command on remot servers

ansible-playbook --inventory myinv playbook.yaml --syntax-check

# Example2:

Install something in linux

Apt-get is a package manager

Git.yaml  
- name: install git client on all vms  
 hosts: prod

**become:** yes ##all tasks below will be run as root user if we write the sudo become here

#become\_user: devops ## now run as this user  
 tasks:

- name: "run apt-get update"

#become: yes

apt:

update\_cache: yes

- name: "install git client"

#become: yes

apt:

name: git

#state: present #installs

state: absent #un-installs

Ansible-playbook --inventory myinv git.yaml

# Variables

Passing value of variable

state: "{{ abc }}" #installs

**Passing value from command line**

Ansible-playbook --inventory myinv Git.yaml --extra-vars "abc=absent"

Ansible-playbook --inventory myinv Git.yaml --extra-vars "abc=present"

**Define value in playbook. This will be over-written by command line value**

- name: install git client on all vms  
 hosts: prod

**become:** yes ##all tasks below will be run as root user if we write the sudo become here

#become\_user: devops ## now run as this user  
 vars:

abc: present

**Both tasks can be clubbed together**

- name: install git client on all vms  
 hosts: prod

**become:** yes ##all tasks below will be run as root user if we write the sudo become here

vars:

abc: present

tasks:

- name: "install/uninstall git client"

apt:

update\_cache: yes

name: git

state: absent #un-installs

Install Tomcat on VMs

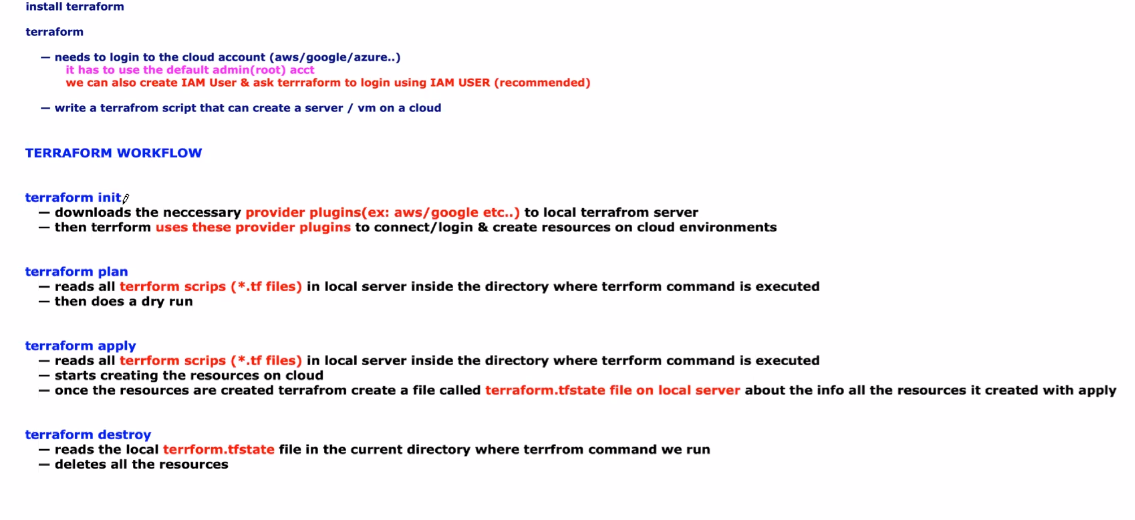
<https://github.com/lerndevops/labs/blob/master/ansible/playbooks/tomcat.yml>

<<tomcat.yml.txt>>

10 - 01 Feb 2023 (Ansible, Terraform)

01 February 2023

20:33



# Ansible Roles

## Meaning:

Multiple plays.

Target execution based on OS type on remote servers.

## Purpose:

Break code in small playbooks, it makes it easy to troubleshoot.

Organize the playbooks properly.

## Ansible Roles:

The role is the primary mechanism fo rb reaking a playbook into multipel files.

It makes it easy to reuse.

Helps to organize a pre-defined framework / directory structure.

Roles helps to share the automation with anyone.

For any help visit

<https://galaxy.ansible.com>

Get role name from galaxy website. From git location.

Absible-galaxy install **leaddevops.jdk**

Create playbook to call the role

-name: install jdk

hosts: prod

become: yes

#tasks:

roles:

leaddevops.jdk

Ansible-playbook -i /home/ubuntu/myinv jdk.yaml

Advanced concept of role in this video

<https://drive.google.com/file/d/1TGxXZqOVi69vXIQye8GNjO4kHfIPlxp9/view?usp=sharing>

# Infrastructure as a Code using Terraform

## Why this culture emerged?

In datacentres creation of server was manual task until few years back.

With infra as a code servers can be provisioned using script.

If we create infra with a script then it is called as infrastructure as a code (IaC).

## Automation tools

**Terraform**

Ansible (it extended its functionalities later)

CloudFormation (AWS Cloud)

Google Deployment Manager (google cloud)

Azure Resource Manager (Azure Cloud)

Terraform can work on any type of cloud like

Alibaba cloud,

Aws cloud,

Google cloud,

Azure cloud,

OpenStack

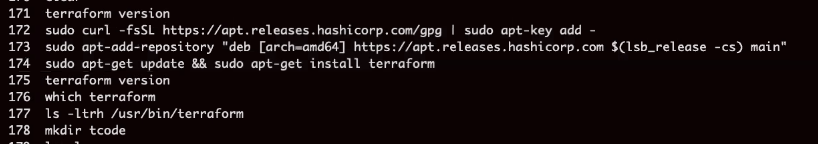
VMWare, many more.

## Install Terraform CLI

Documentation:

<https://github.com/lerndevops/labs/tree/master/terraform/install>

|  |
| --- |
| **Install Using Script**  sudo wget <https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installTerraform.sh> -P /tmp sudo chmod 755 /tmp/installTerraform.sh sudo bash /tmp/installTerraform.sh  **Manual Installation Steps**  **Install Terraform CLI on Ubuntu/Debian Linux OS**  sudo curl -fsSL <https://apt.releases.hashicorp.com/gpg> | sudo apt-key add - sudo apt-add-repository "deb [arch=amd64] <https://apt.releases.hashicorp.com> $(lsb\_release -cs) main" sudo apt-get update && sudo apt-get install terraform  Verify : terraform version  **Install Terraform CLI on CentOS/RHEL Linux OS**  sudo yum install -y yum-utils sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo> sudo yum -y install terraform  Verify : terraform version |



Terraform plugins are called providers

Use providers

Make a call to cloud

It has to be authenticated.

To download plugin write a simple script.

Mkdir tcode

Cd tcode

Vi provider**.tf**

Provider "aws" {

}

Provider "google" {

}

Provider "azurerm" {

}

Cat prider.tf

Terraform init

It automatically readers provider.tf file.

Initializes provider plugins.

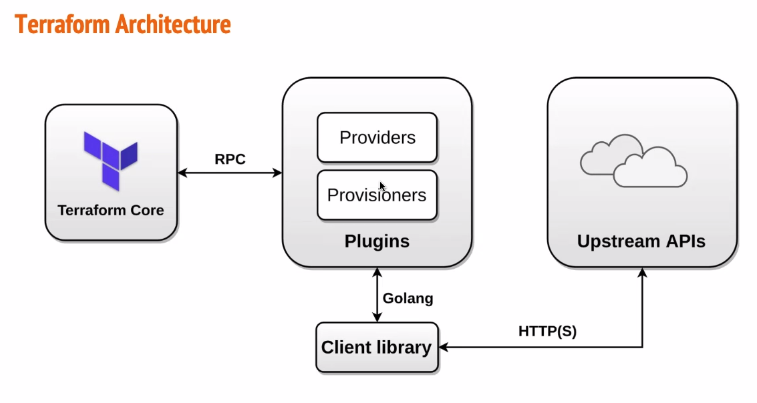
It simply downloads all plugins.

Where it downloads - incurrent directory. Puts in **.terraform** directory.

By default it connects to online registry

It downloads it from <https://registry.terraform.io>

Terraform init requires internet connection to download plugins.



**How to write Terraform script?**

HCL - HashiCorp configuration language

TCL - Terraform Configuration Language

**Terraform Terminology**

Provider - it is a cloud provider name, i.e., plugins

Resources - on any cloud anything we create is called resource.  
like container, vm, etc.

All the terraform scripts must end with\*.tf extension. The file name can be anything.

Arguments = scalar

are always **key = value** pair

Blocks = dictionary\map

Is set of instructions enclosed in curly braces

Usually all the arguments are enclosed wihin a block.

blocktype{

Intance\_type = t2.micro

Key\_name = edujan16.pem

}

**Different block types we use to create infra needed**

Providers / resources / provisioner / vars

## Create a script

Vi ec2.tf

resource "aws\_instace" "dev" {

tags = {

#create this in aws launch instance page

Name = "devinstance1"

env = "Dev"

}#tags

ami = "ami-……." #copy it from aws portal. Launch instance page

instance\_type = "t2.micro"

key\_name = "edujan16"

secutiry\_groups = ["edujan16","securitygroup2"] #we can give multipel secutiry group names

}#resource

## Execute command

#This will check read .tf files and tell us if ther is nay sytax errror or not]

Terraform plan

#Execute .tf file

Terraform init

Sign in page

IAM user

Select root user

Search --> IAM

#this will allow to create user and password

Users -> add user

Username = terra

Continue

Permission

Attach existing policies

Search -> ec2full

AdministratorAccess

Next

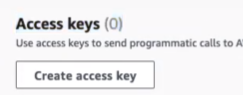
Create user

Open User terra ->

security Credentials

Access keys

Create access key



Command line interface (CLI)

Terraform

Next

Name = terra

Copy access key (username) and secret access key (password)

Download .csv file

**Open provider.tf**

Provider "aws" {

Access\_key = "adsfdfsdf"

Secret\_kehy = "adfdsfdsf"

Region = "us-east-2"

}

Access Key: AKIAQCR47G277TBNDCY3

Secret: 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

**Check again**

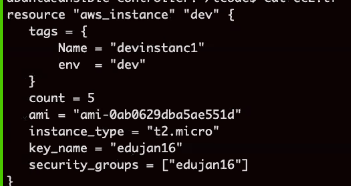
Terraform plan

**Execute code**

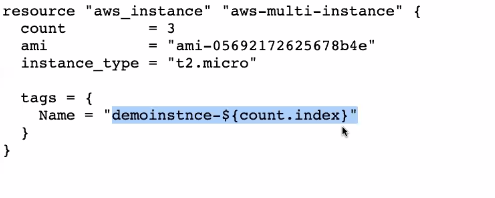
Terraform apply

## Want to add more instance

Count = 5



## Different name for each VM



## Delete VMs

Terraform destroy

11 - 02 Feb 2023 (Ansible)

01 February 2023

20:34

Agenda

Ansible (continue….)

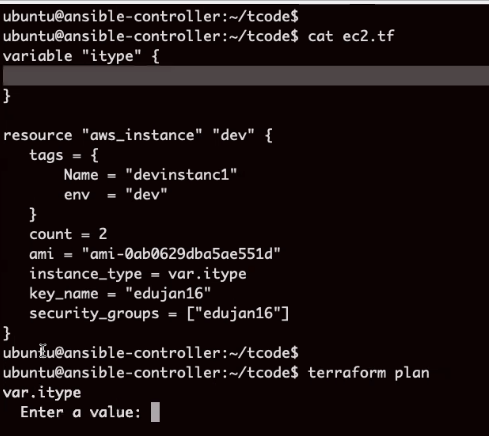
Container

# Ansible

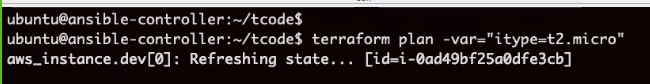
## Define a variable in terraform file

## 

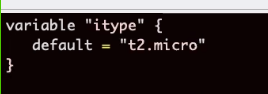
## Variable value is blank in script



## Pass it from command line

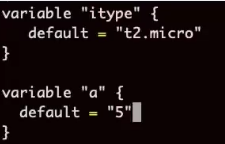


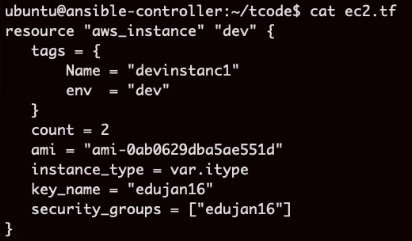
## Set default value in script itself



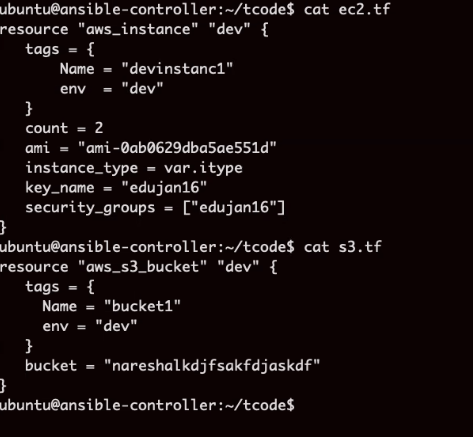
## Save all variables in a separate file

**variables.tf #file name should be this only. Cannot change file name.**





Create bucket in AWS. Bucket is used to store files nd fodlers so that it can be shard with team members. Equal to google drive



We have been using all pre-defined key-words.

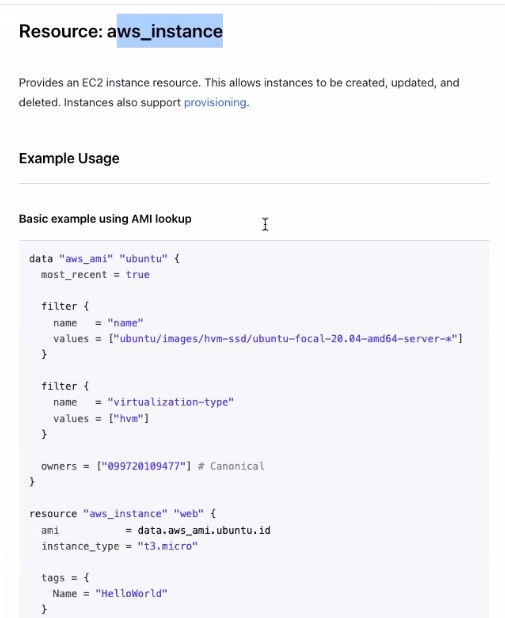
Where to find these key-words

Visit: Aws/latest/docs



Search for compute or ec2 instance

Aws\_instance



**Show just resource names created using terraform**

Terraform state list

**Just show particular resource detail**

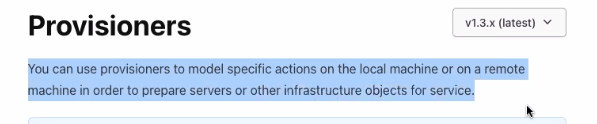
Terraform state show aws\_s3\_bucket.dev #resource name

**Terraform destroy** reads terraform.state file and destroy all resources

**Delete particular resource**

Terraform destroy -target <resourcename>

Terraform provisioners



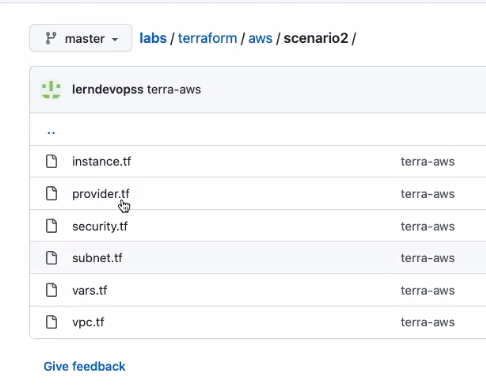
**Provisioner & trigger ansible playbook via Terraform**



Trigger



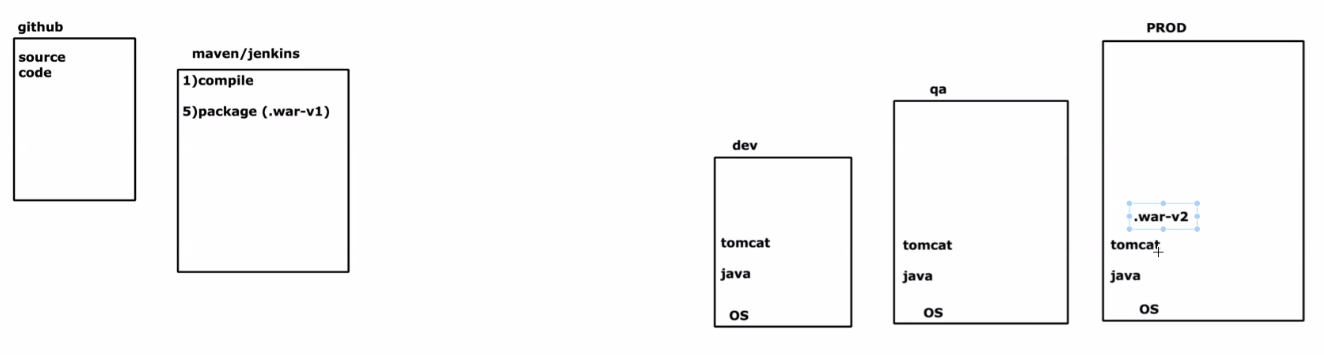
## Assignment: Bigger project



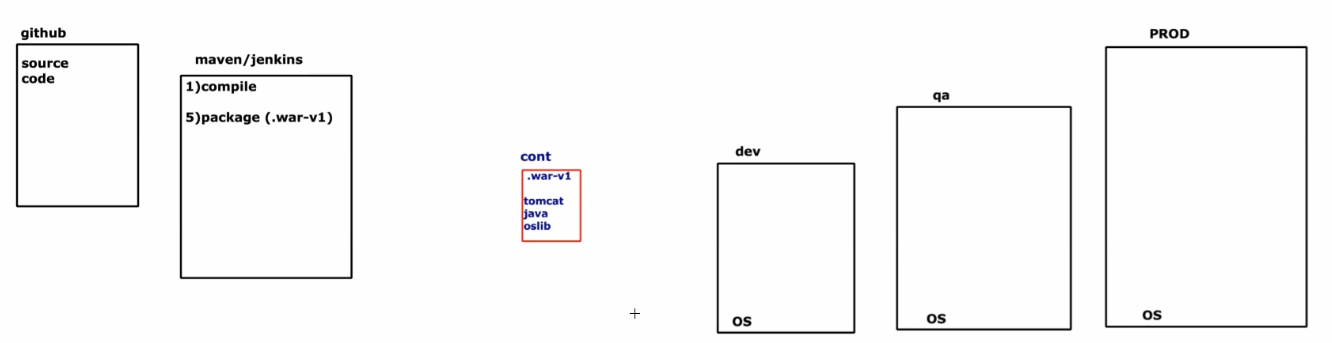
# Containerization

## Meaning

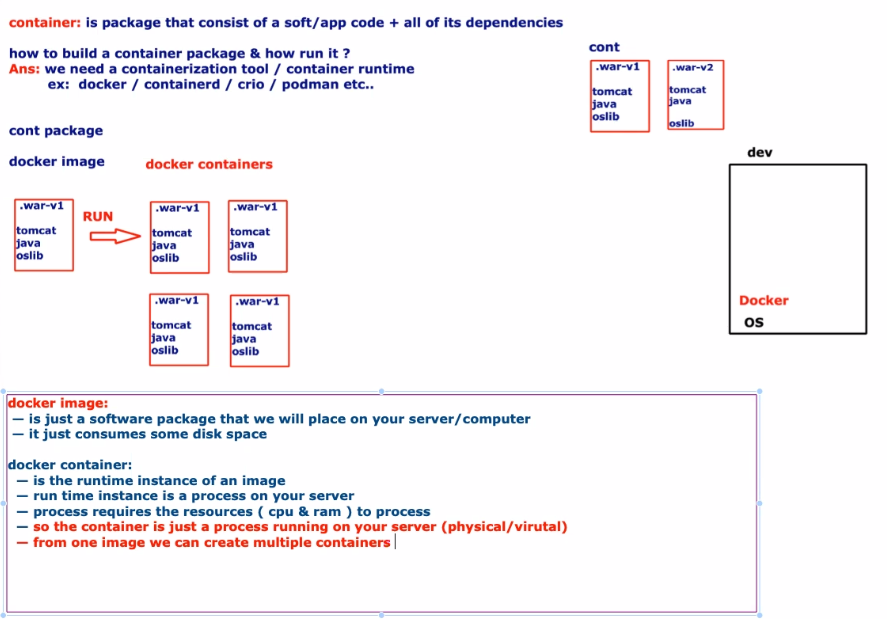
Review of what we did till now



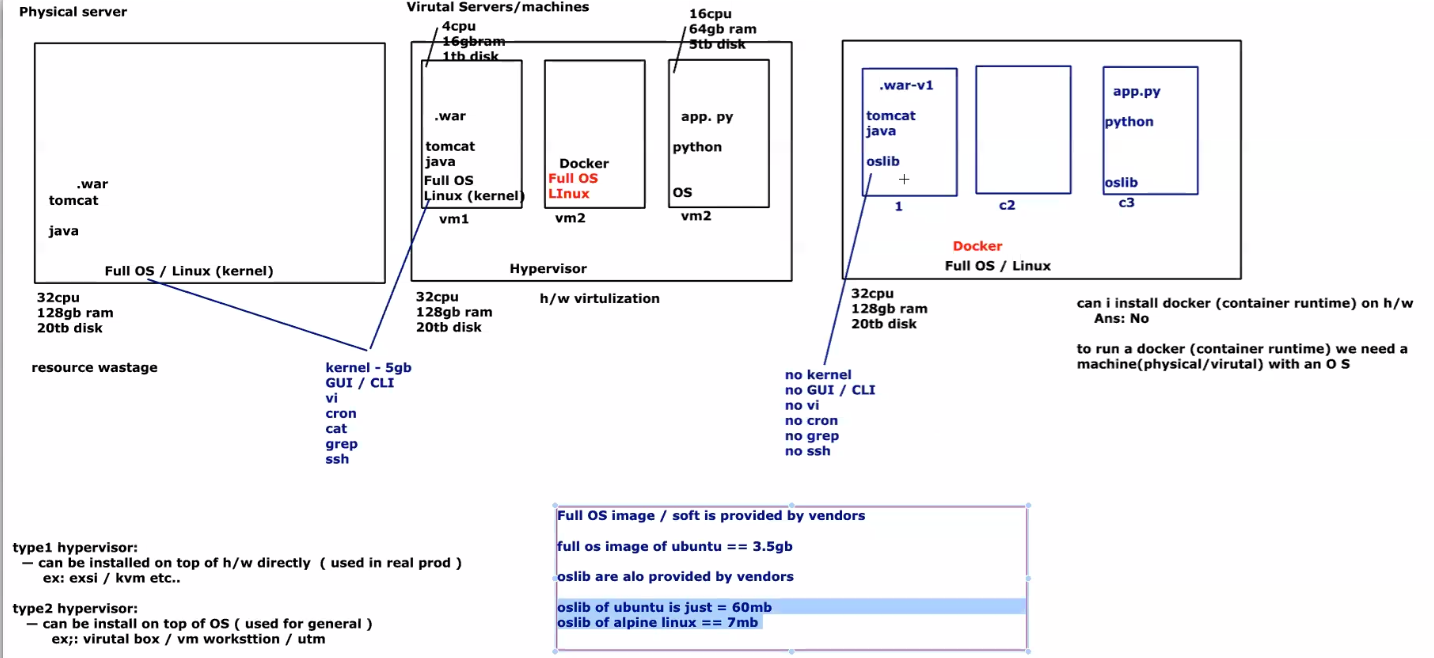
## With Container



## More definition



## Meaning of Container - difference between container and full Operating system



**Install Docker**

<https://github.com/lerndevops/labs/tree/master/docker/install>

|  |
| --- |
| **Install Using Script**  sudo wget <https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh> -P /tmp sudo chmod 755 /tmp/installDocker.sh sudo bash /tmp/installDocker.sh |

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gnupg software-properties-common lsb-release

## Add Dockers official GPG key & stable repo

sudo mkdir -p /etc/apt/keyrings

curl -fsSL <https://download.docker.com/linux/debian/gpg> | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] <https://download.docker.com/linux/debian> $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

## Install Docker latest

sudo apt-get update ; clear

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

## Test the installation by running a simple container:

docker run hello-world

## FYI Only ## Check Available Version for Install

apt-cache madison docker-ce | awk '{print $3}'

**Install location:** cd /var/lib/docker