
AUTOMATING SERVER SETUP WITH
ANSIBLE

CONTENT OF TABLES

CONTENT OF TABLES

CONTENT OF TABLES

CONTENT OF TABLES

CONTENT OF TABLES

Ansible Labs and Used Modules :

- Ansible Run Any Scripts on Hosts
- Ansible Shell and Script and Command module and Compares
- Ansible DataBase Modules
  - Create Databases
  - Create Users and Password
  - Create PRIVILEGES and Grant
  - Import into DataBase
- Create Variable User_List in vars Directory and Used in to task Directory
- Ansible Notifications Handlers
- Ansible Waiting for events and search_regex
- Create a to-do-list with include tasks Module
- Using filters with conditionals and using When Module
- Insert to File Module
- Find Module in any directory and Deleted and using wildcard

CONTENT OF TABLES

CONTENT OF TABLES

Install Project

- Install Apache
- Install Nginx
- Install Mariadb and import Databases
- Change Password for all hosts
- Install zabbix Agent
- Install VSFTP with htpasswd
- Install zabbix server with Compile Model (Optional)

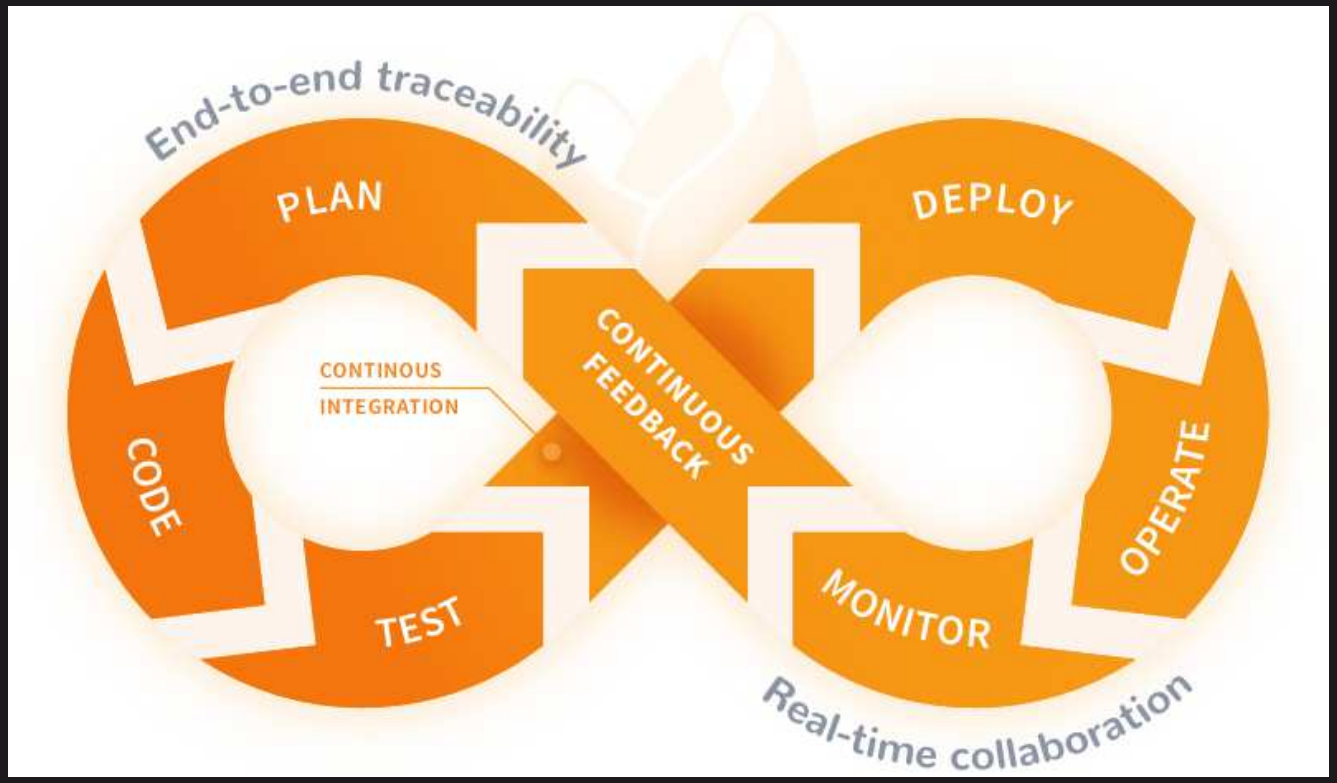
INTRODUCTION DEVOPS AND DEVSECOPS

WHAT IS DEVOPS?

IT'S NOT A TEAM OR TITLE … IT'S A PROCESS.
BECAUSE CUSTOMER OF THE KINGS

INTRODUCTION DEVOPS AND DEVSECOPS

DevOps Lifecycle



INTRODUCTION DEVOPS AND DEVSECOPS

DevOps Timeline

2007 — Patrick Debois start's assessing IT Value Chain

2008 — Agile System Administrators Group is launched on Google

Google

Agile2008 Conference

Andrew Clay Shafer and Patrick Debois meet at Agile Conference 2008

2009 — Inaugural "DevOps Days" are held in Ghent, Belgium

DEVOPS

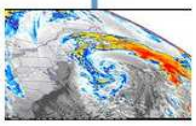
CHEF

Open Source toolsets rip up the legacy playbook.

O'REILLY Velocity Web Performance & Operations CONFERENCE

John Allspaw and Paul Hammond present "10 Deploys per day" at Velocity

2010



The "Perfect Storm" of adjacent methodologies occurs

2011 — Industry leading software vendors increase market presence with "Enterprise" class DevOps tools

IBM

hp

ca technologies

Industry leading software vendors increase market presence with "Enterprise" class DevOps tools

2012

Gene Kim releases "The Phoenix Project"

2013

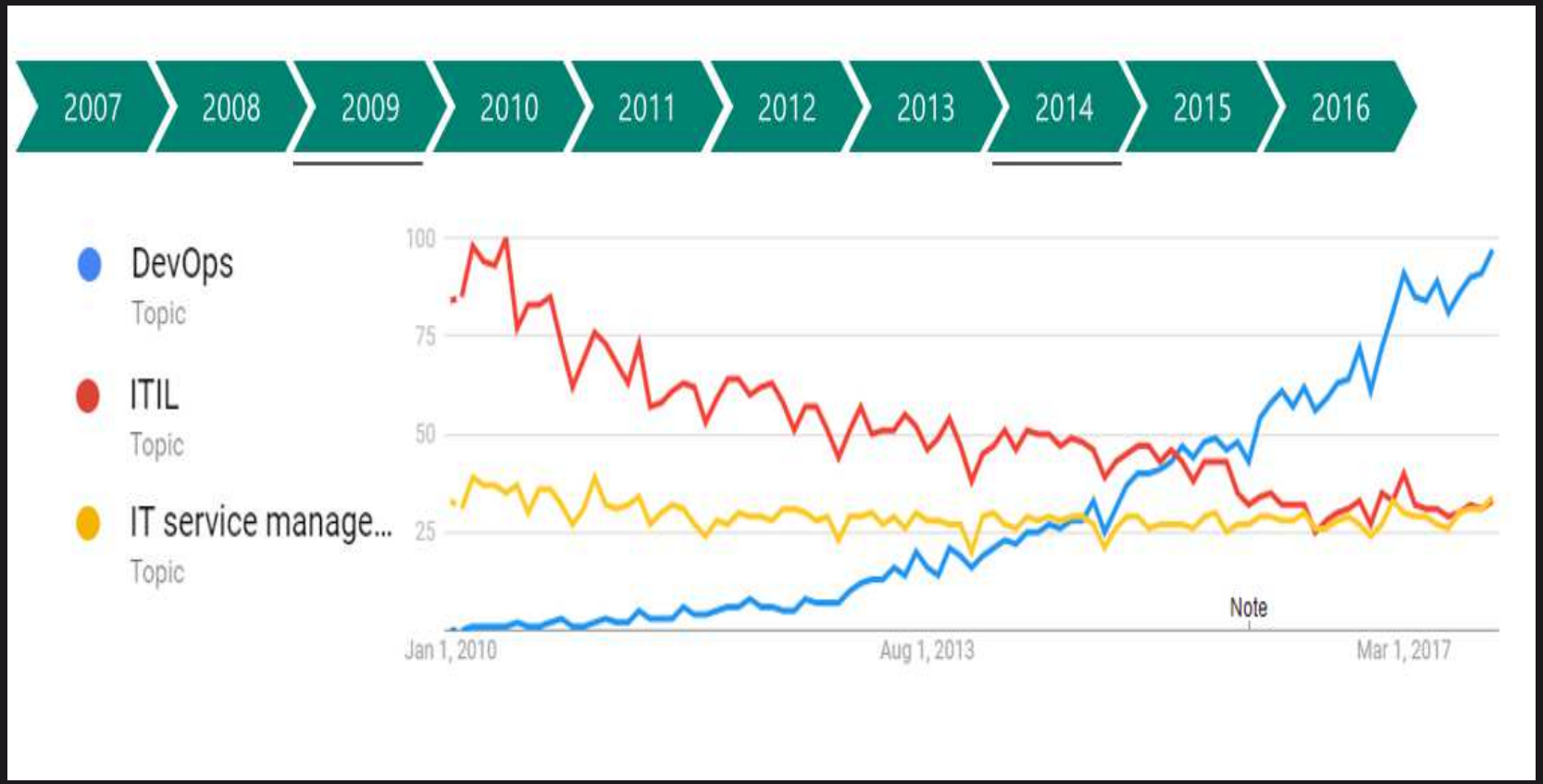
DevOps begins to provide positive impact to "Enterprise" IT

2014 — *devX* is born and Xceed launches the "12 days of DevOps"

2015

What does the future of DevOps hold for your organisation?

INTRODUCTION DEVOPS AND DEVSECOPS



INTRODUCTION DEVOPS AND DEVSECOPS

WHAT IS DEVSECOPS ?

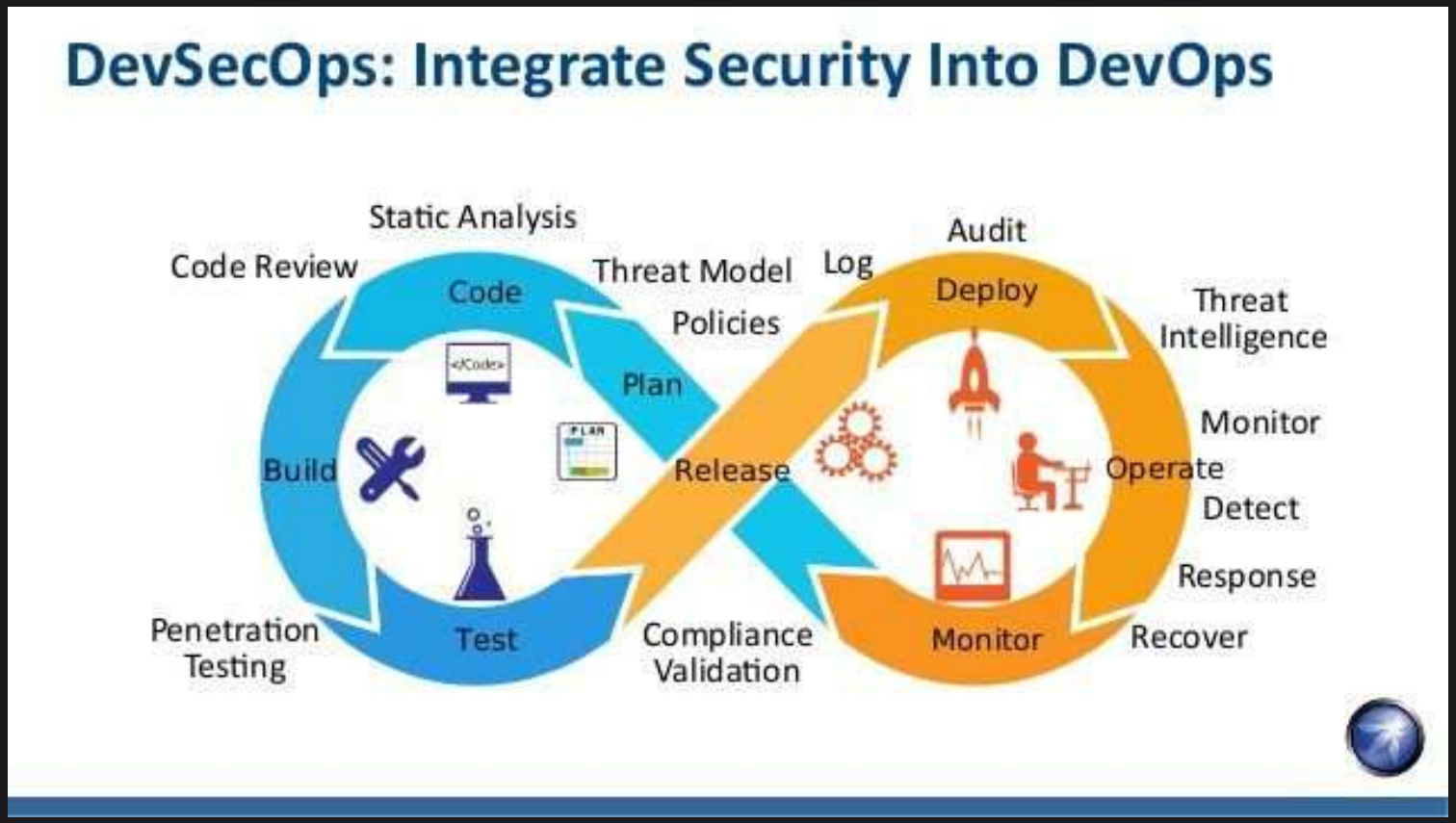
IT IS SHORT FOR :

DEVELOPMENT

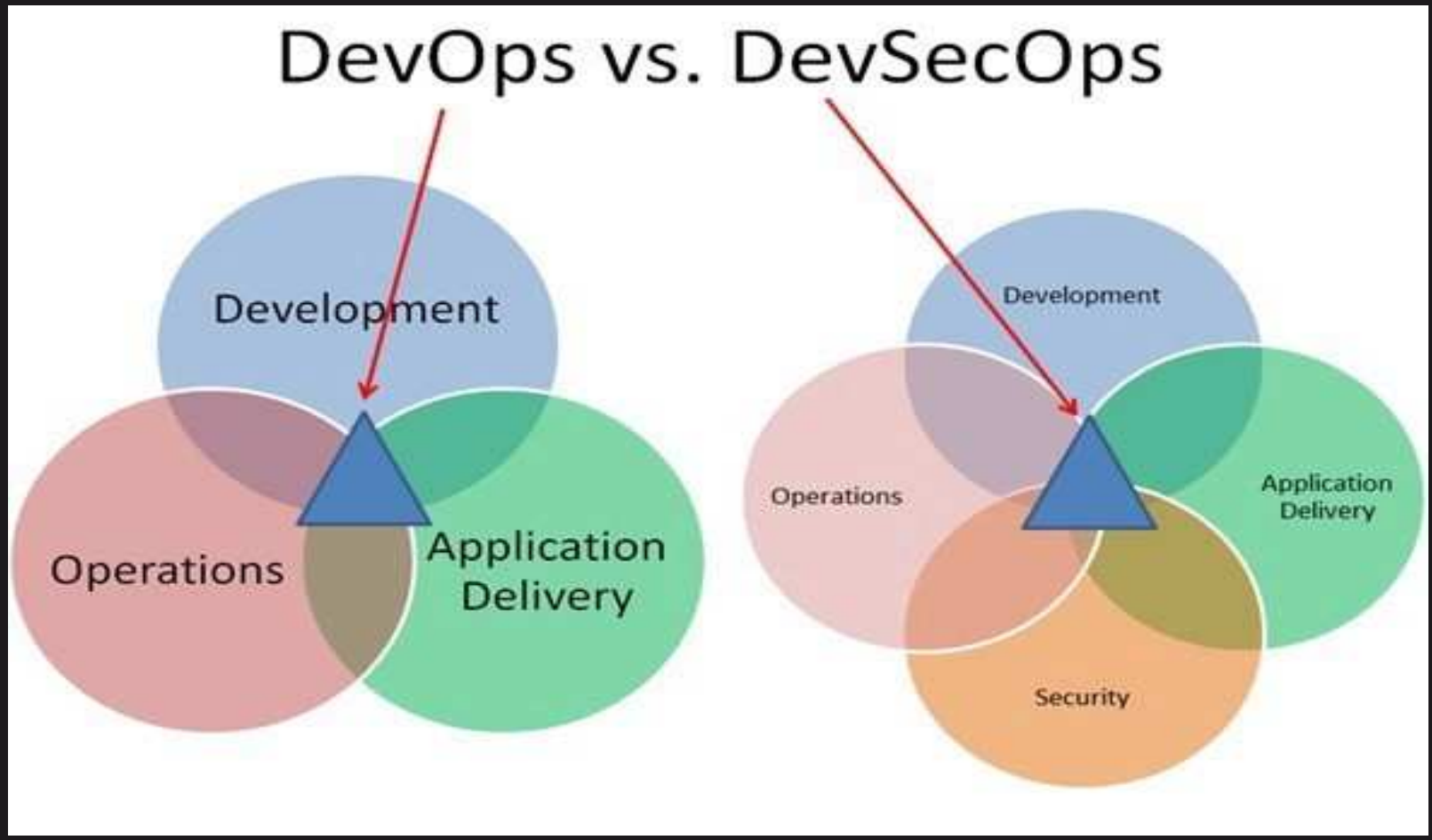
SECURITY

OPERATIONS

INTRODUCTION DEVOPS AND DEVSECOPS



INTRODUCTION DEVOPS AND DEVSECOPS



INTRODUCTION DEVOPS AND DEVSECOPS

TARGET :

New versions of the product, very Quick to final Customers

Improve Speed to Market

PROBLEM :

Seeing any Problem in the operation

Each Team attributed it to the Other Team

SOLUTION :

Collaboration

Remove the wall between Operation and Developer

DevOps Engineering Tools

BENEFITS :

TrubleShooting Speed

Isolation Environment

Performance Tuning

Fail Over

Resource & Cost Reduction

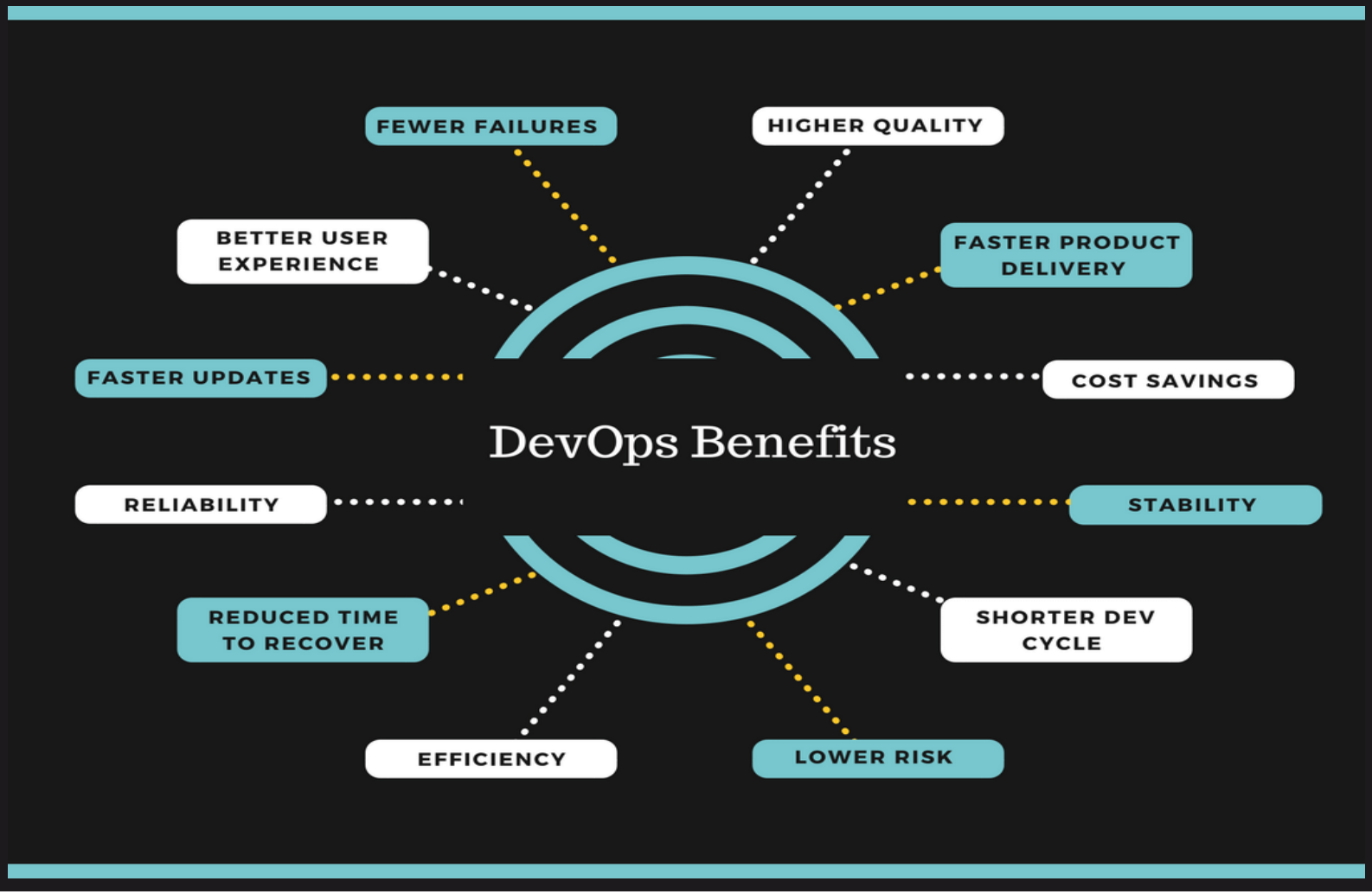
Increased Performance

Job Satisfaction

RESULT :

The transformation of the IT industry

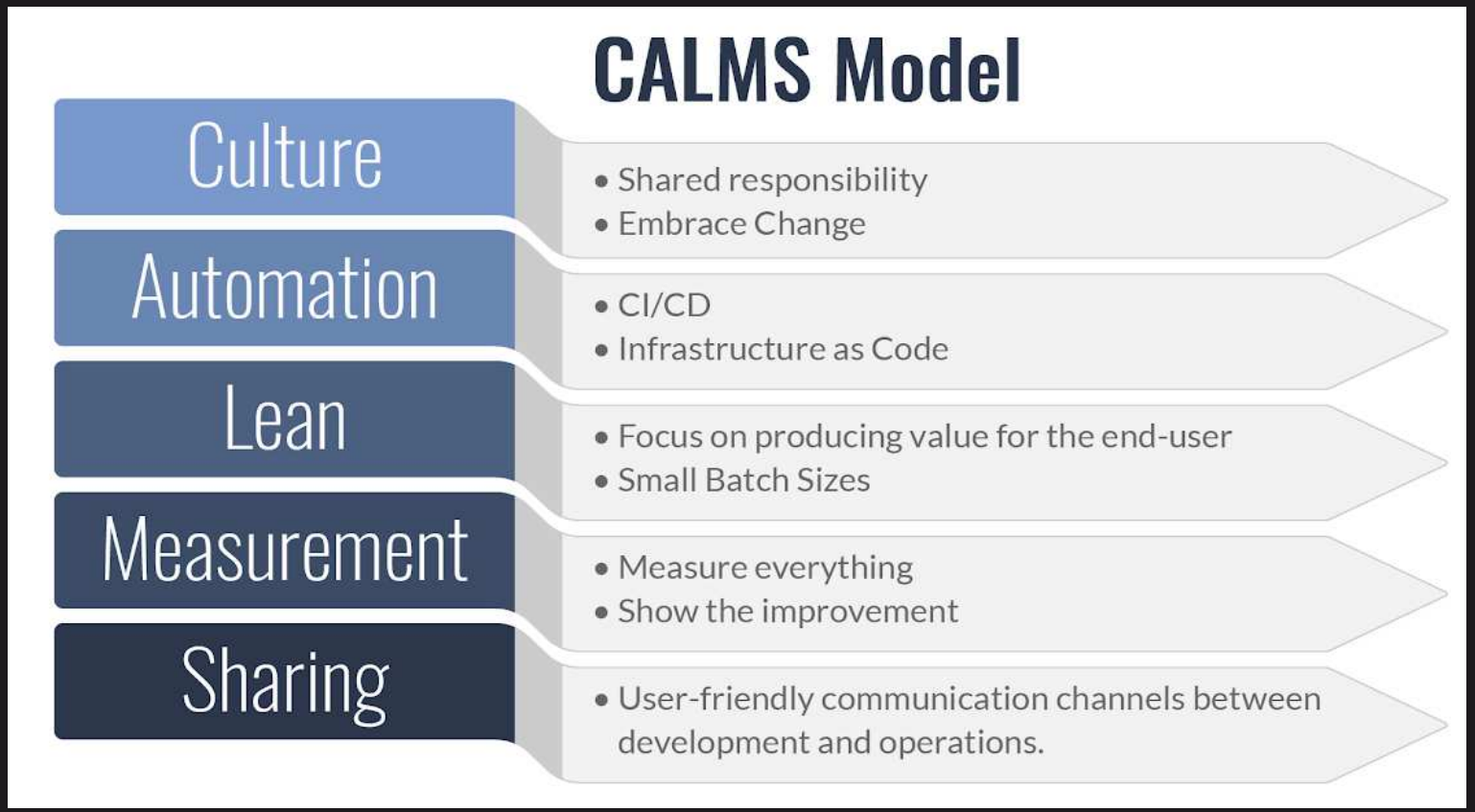
INTRODUCTION DEVOPS AND DEVSECOPS



INTRODUCTION DEVOPS AND DEVSECOPS

DevOps is Process

Approach to reach DevOps with CALMS



INTRODUCTION DEVOPS AND DEVSECOPS

DevOps is Process

Approach to reach DevOps with CALMS

Culture : Delete Wall between Developers Team and Operations Team

Automation : Continuous Delivery – Continuous Integration – Continuous Deployment

Use tools (Configuration Management and Virtualization , …)

Lean : Delete Redundant work and Useless

Measurement : Unless we know where we are, we will not know where we want to go

Infrastructure Monitoring

Log Management

Application and Performance Management

→ TPS and Response Time , …

Sharing : Share Information and Result with Coworker

DevOps Use cases :

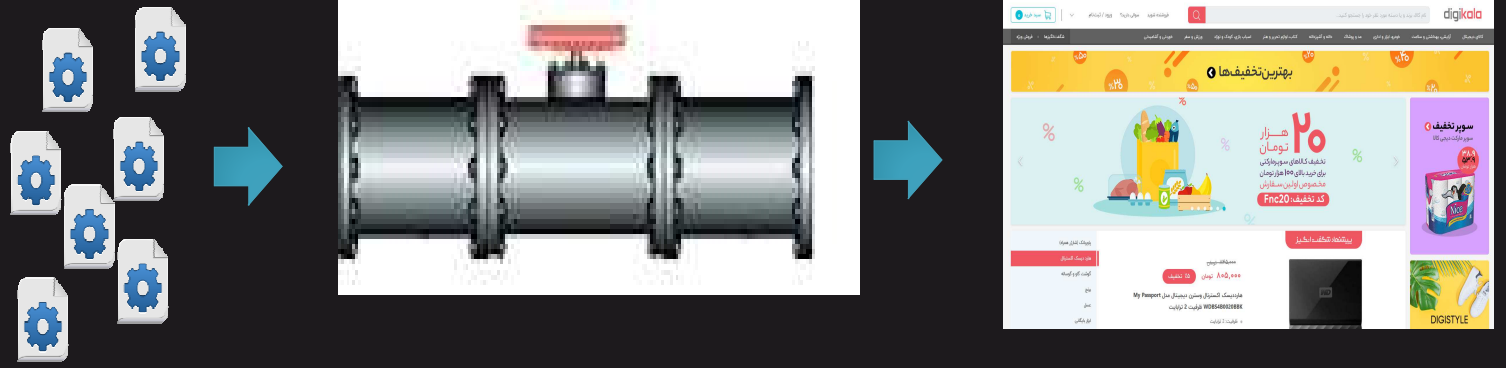
High changes in product

Add More Features

Competitive products

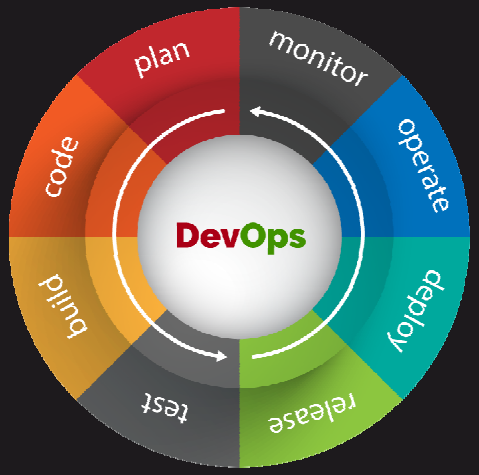
INTRODUCTION DEVOPS AND DEVSECOPS

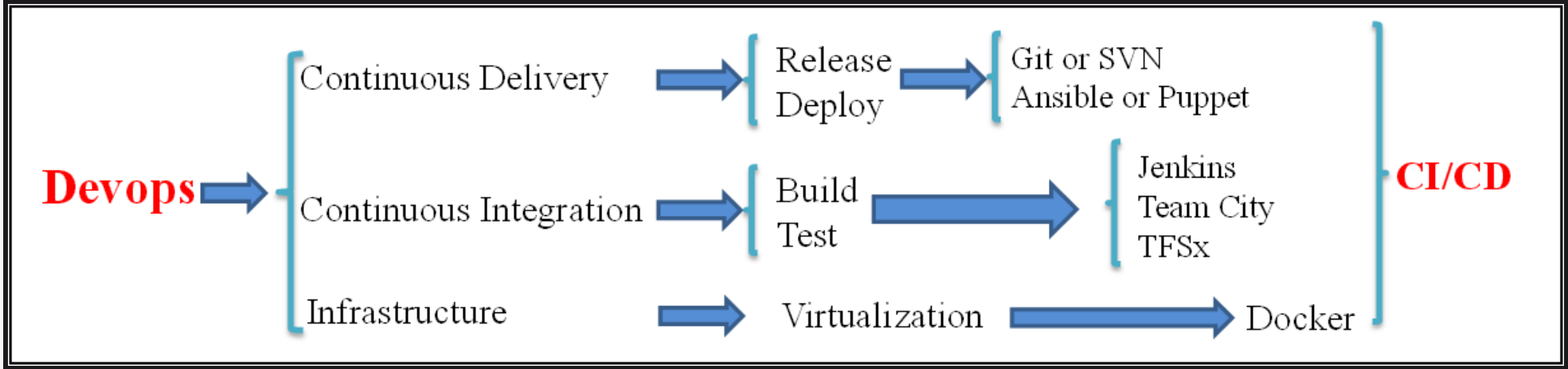
DevOps Pipline



INTRODUCTION DEVOPS AND DEVSECOPS

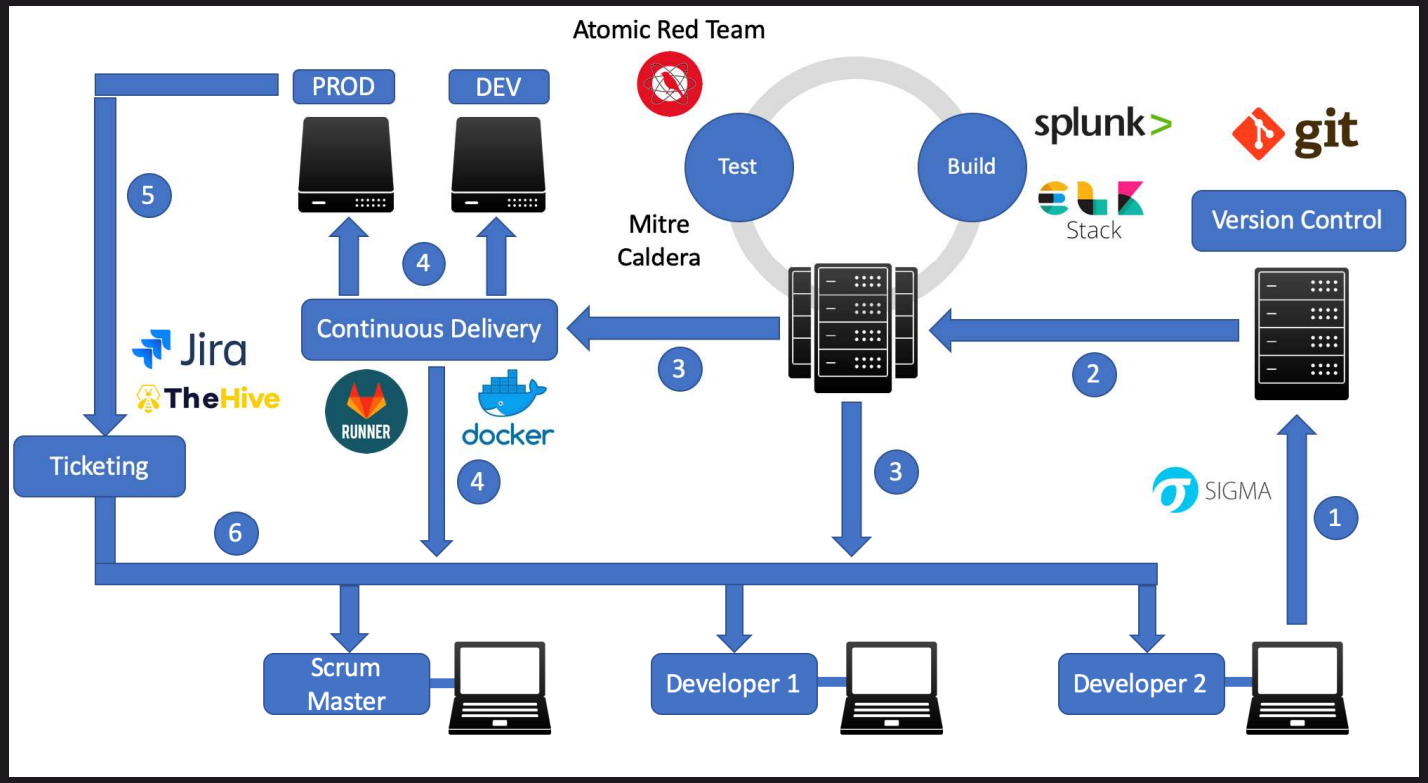
DevOps Pipline





INTRODUCTION DEVOPS AND DEVSECOPS

DEVSECOPS PIPELINE IS COMPLETED

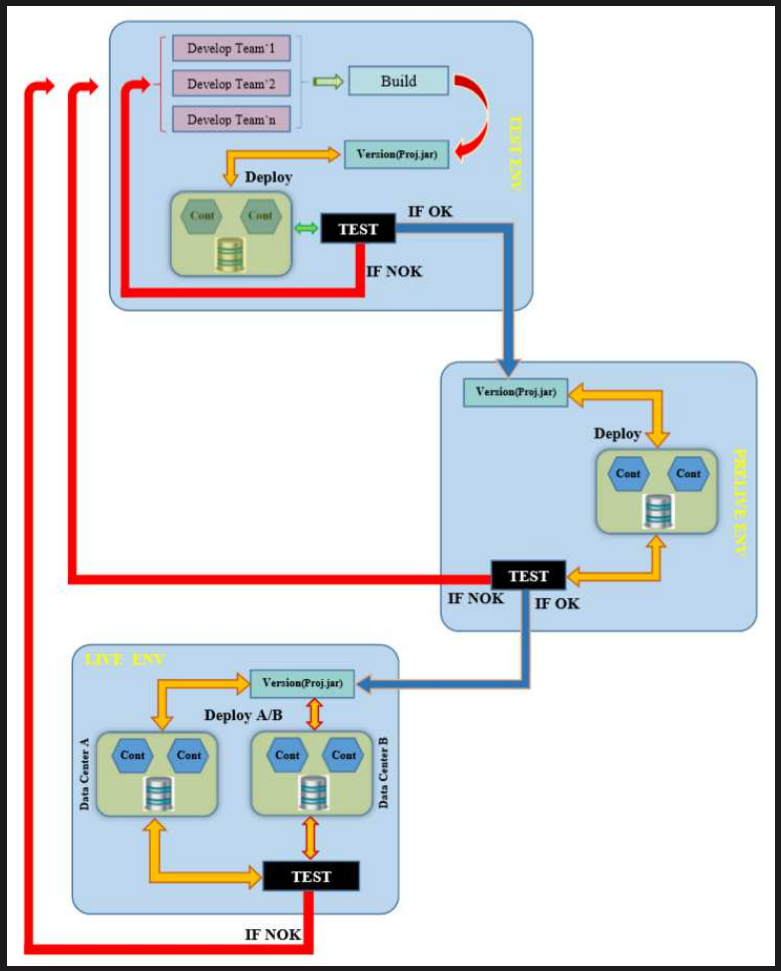


INTRODUCTION DEVOPS AND DEVSECOPS

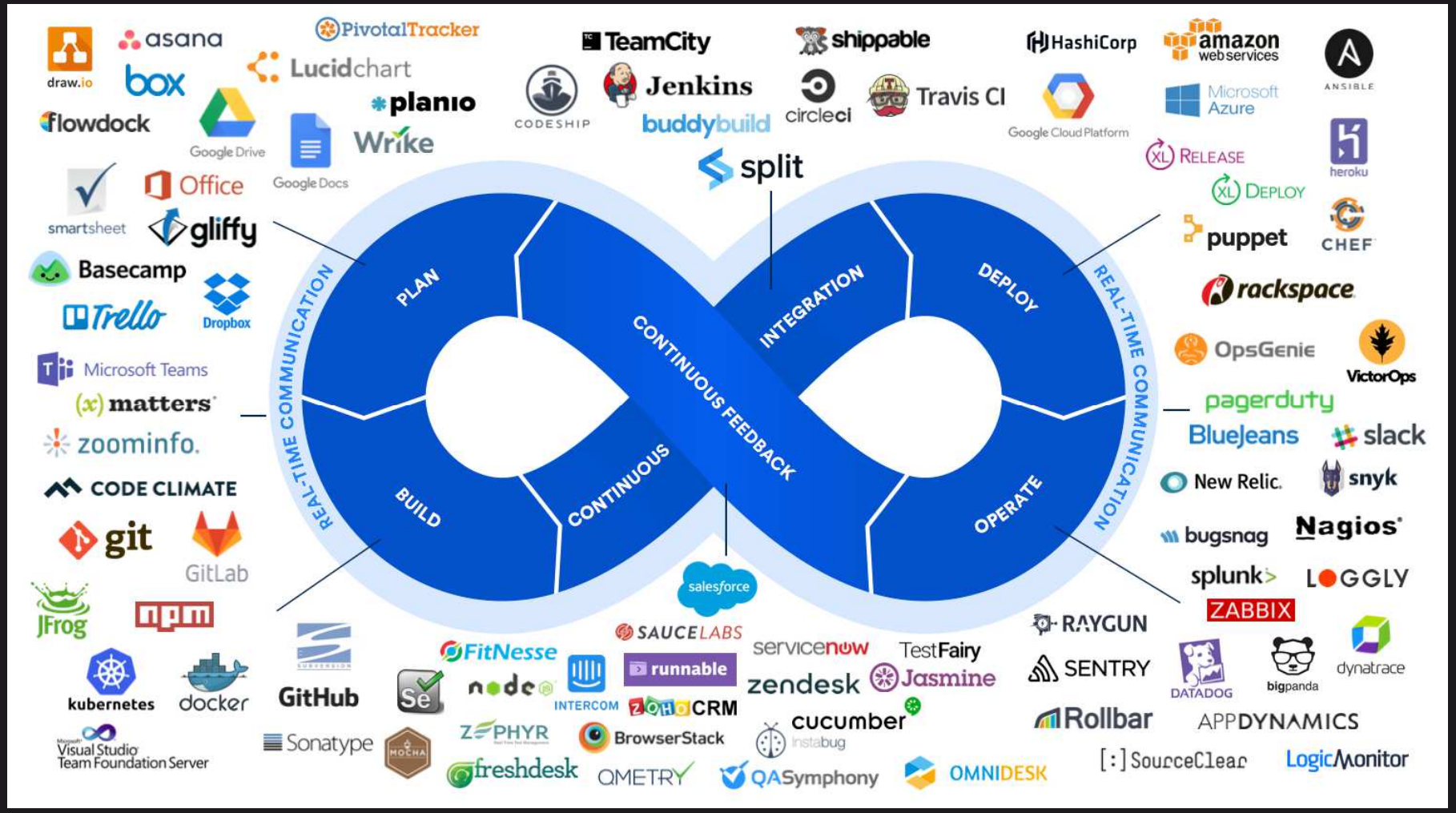
Building Application Deployment Pipelines and Environments

Environment :

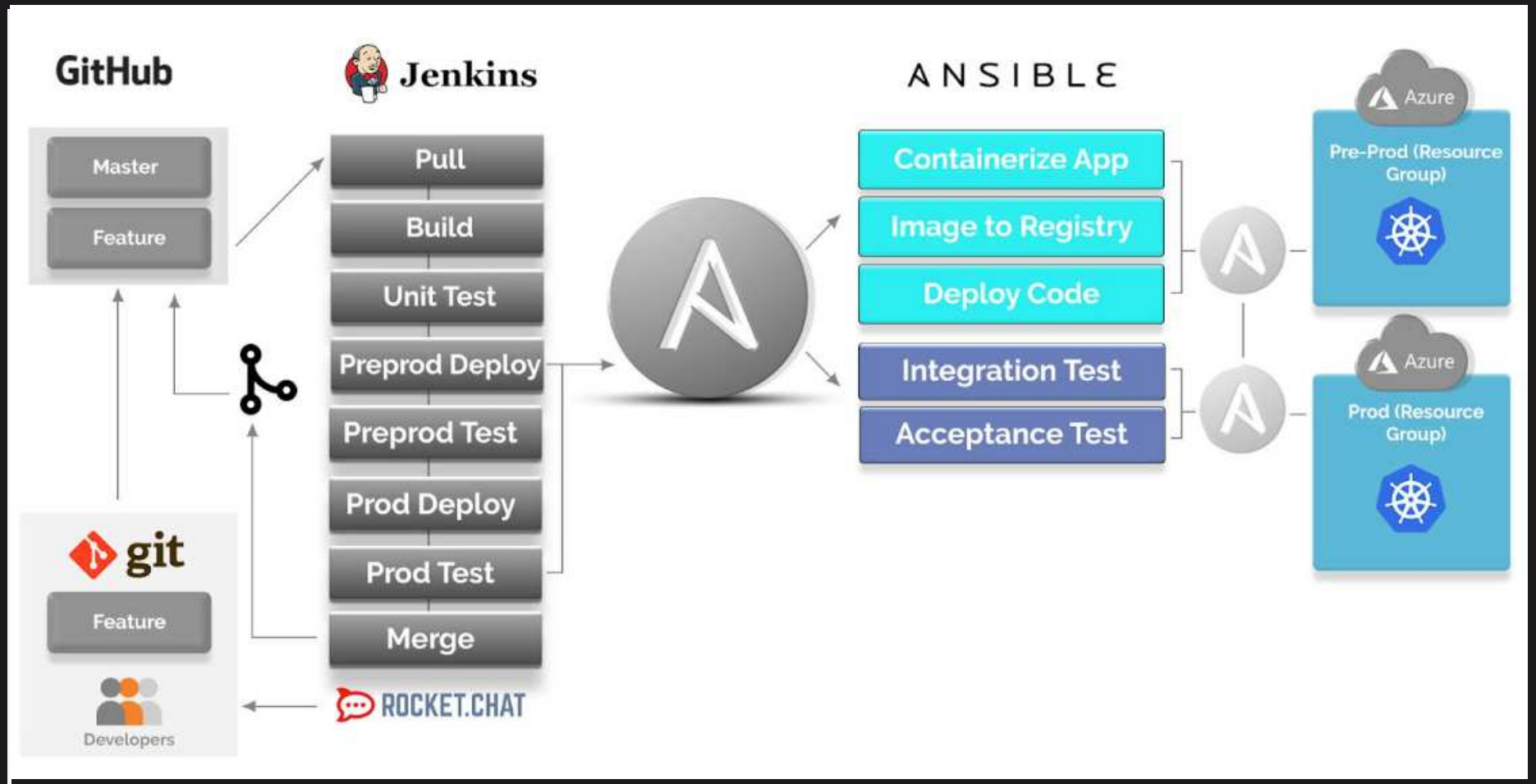
- Test Env
- Prelive Env
- Live Env



DEVOPS ENGINEER TOOLS



CONFIGURATION MANAGEMENT ROLE IN CI/CD ARCHITECTURE



CONFIGURATION MANAGEMENT AND ORCHESTRATION ROLES IN DEVOPS

Configuration Management in DevOps is

coined as "comprehensive configuration management" and is made up of

1- Source Code Repository : only Development phase

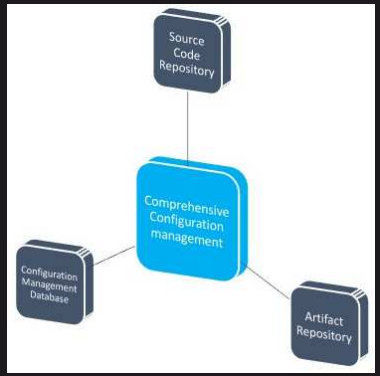
the source code repository is a database of source code which developers use.

This database serves as :

A container for all the working code

Source code aside

It stores a number of useful components including various scripts and configuration files.



2- Artifact Repository : Development and Operation phase

An artifact repository is meant to store machine files.

This can include :

binaries files

test data files

libraries files . Effectively, it’s a database for files that people don’t generally use.

3- Configuration Management Database : Development and Operation phase

configuration management database or (CMDB) is a relational database multiple systems and applications related to configuration management

including services, servers, applications, and databases to name a few.

INFRASTRUCTURE ENVIRONMENT AS A CODE TO MARKET

Results of Properly Managed Configurations

1- Infrastructure-as-a-Code : (Install)

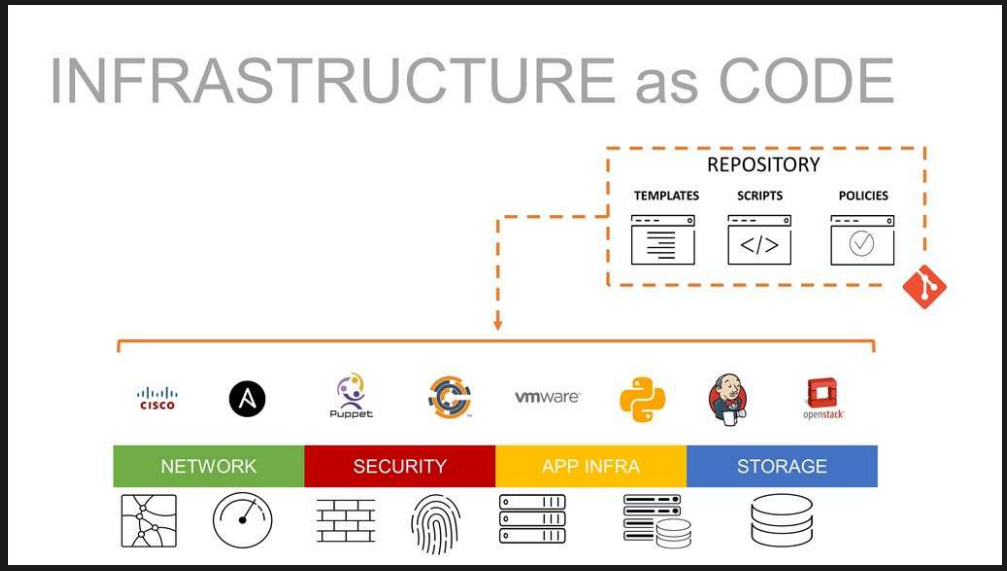
Automatically Create Environments and Infrastructures

Include :

servers , networks of configurations and other resources.

For Example :

Install and Configure Kubernetes Cluster with Configuration Management



INFRASTRUCTURE ENVIRONMENT AS A CODE TO MARKET

Results of Properly Managed Configurations

2- Configuration-as-a-Code : (Change and Deployment Configuration)

Automatically Change management and Central Configuration and Delpoyment Application

Include :

servers , networks of configurations and other resources.

For Example :

CI/CD

INFRASTRUCTURE ENVIRONMENT AS A CODE TO MARKET

Results of Properly Managed Configurations

3- Benefits of IaaC and CaaC

- Automation of the infrastructure environment provides standardization
- Setups are free of human error
- Collaboration is enhanced between operations and development
- Makes infrastructure more flexible, ready to scale
- Each step is consistent across all resources
- Version control is a given

INTRODUCTION TO ANSIBLE

ANSIBLE IS IT AUTOMATION PLATFORM THAT MAKES YOUR APPLICATIONS AND SYSTEMS EASIER TO DEPLOY.

IT SUPPORT CONFIGURATION MANAGEMENT WITH EXAMPLES AS BELOW.

- INSTALL AND CONFIGURATION OF SERVERS
- APPLICATION DEPLOYMENT
- CONTINUOUS TESTING OF ALREADY INSTALL APPLICATION
- ORCHESTRATION
- AUTOMATION OF TASKS

WHY ANSIBLE

Ansible Configuration Management For Automation :

- It is a free open source application
- Agent-less – No need for agent installation and management
- Phython/yaml based
- Highly flexible and configuration management of systems.
- Large number of ready to use modules for system management
- Custom modules can be added if needed
- Configuration roll-back in case of error
- Simple and human readable
- Self documenting

WHY AUTOMATION?

- Tasks in code
- Collaboration
- Eliminate errors
- Write once
- Laziness
- Etc….

ANSIBLE VS. OTHER CM TOOLS

The Best DevOps tools for 2019

- ✓ **Configuration Management Tools :**
  - ❑ **CFEngine**
  - ❑ **Puppet :** master-slave architecture.
  - ❑ **CHEF**
  - ❑ **Ansible**
  - ❑ **Salt**

LPI Exam 701: DevOps Tools Engineer

Topic 704: Configuration Management

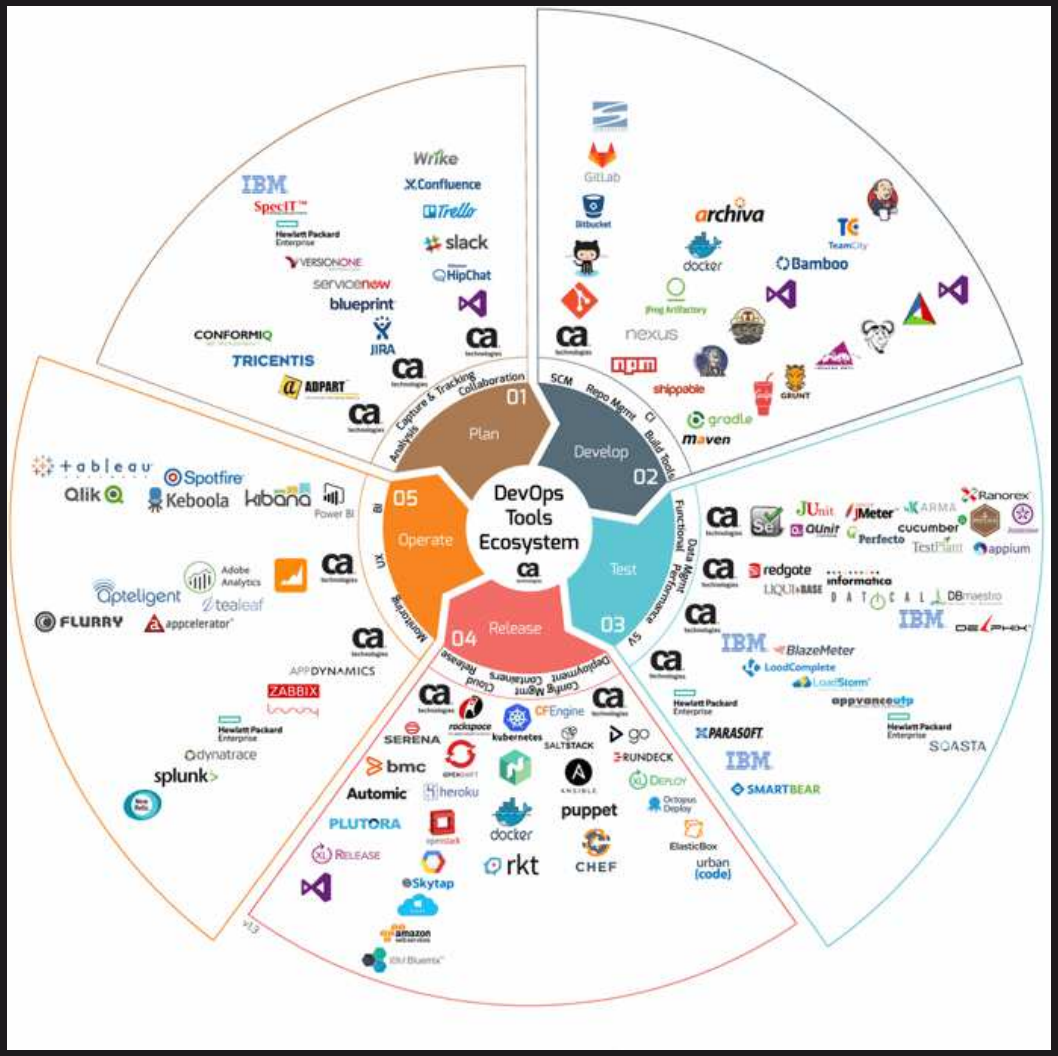
Ansible : ansible.cfg , ansible-playbook , ansible-vault , ansible-galaxy , ansible-doc

Other Configuration Management Tools :

CHEF , Puppet

	Language	License	Mutual auth	Encrypts	Verify mode	Agent-less	Have a GUI	First release	Latest stable release
Ansible	Python	GPLv3+	Yes[1]	Yes[2]	Yes	Yes	Yes[3]	2012-03-08	2018-12-13 2.7.5[4][5][6]
Chef	Ruby, Erlang	Apache 2.0	Yes[14]	Yes[15]	Yes[16][17]	No	Yes	2009-01-15 0.5.0	2019-01-28 14.10.9 (client),[18] 2018-02-13 12.17.33 (server)[19]
CFEngine	C[20]	GPLv3[21]	Yes[1]	Yes[22]	Yes[23][24]	No	Yes[25]	1993	2019-07-01 3.14.0,[26] 2019-05-20 3.12.2,[27] 2019-05-10 3.10.6[28]
Puppet	C++ & Clojure from 4.0,[45] Ruby before then	Apache from 2.7.0, GPL before then	Yes[46]	Yes[9]	Yes[47][48]	No	Yes[49]	2005-08-30[50]	2019-01-15 6.0.5[51]
Salt[72]	Python[73]	Apache 2.0[74]	Yes[75]	Yes[75]	Yes	Both[76][77]	Yes[78][79]	2011-03-17 0.6.0[80]	2019-02-25 v2019.2.0[81]

ANSIBLE IN DEVOPS



INTRODUCTION TO YAML

YAML includes a markup language with important construct

The Design Goals and features of YAML are given below :

- Matches native data structures languages such as Perl, Python, PHP, Ruby and JavaScript
- YAML data is portable between programming languages
- Includes data consistent data model
- Easily readable by humans
- Supports one-direction processing
- Ease of implementation and usage

ANSIBLE PROVISION

Provisioning means :

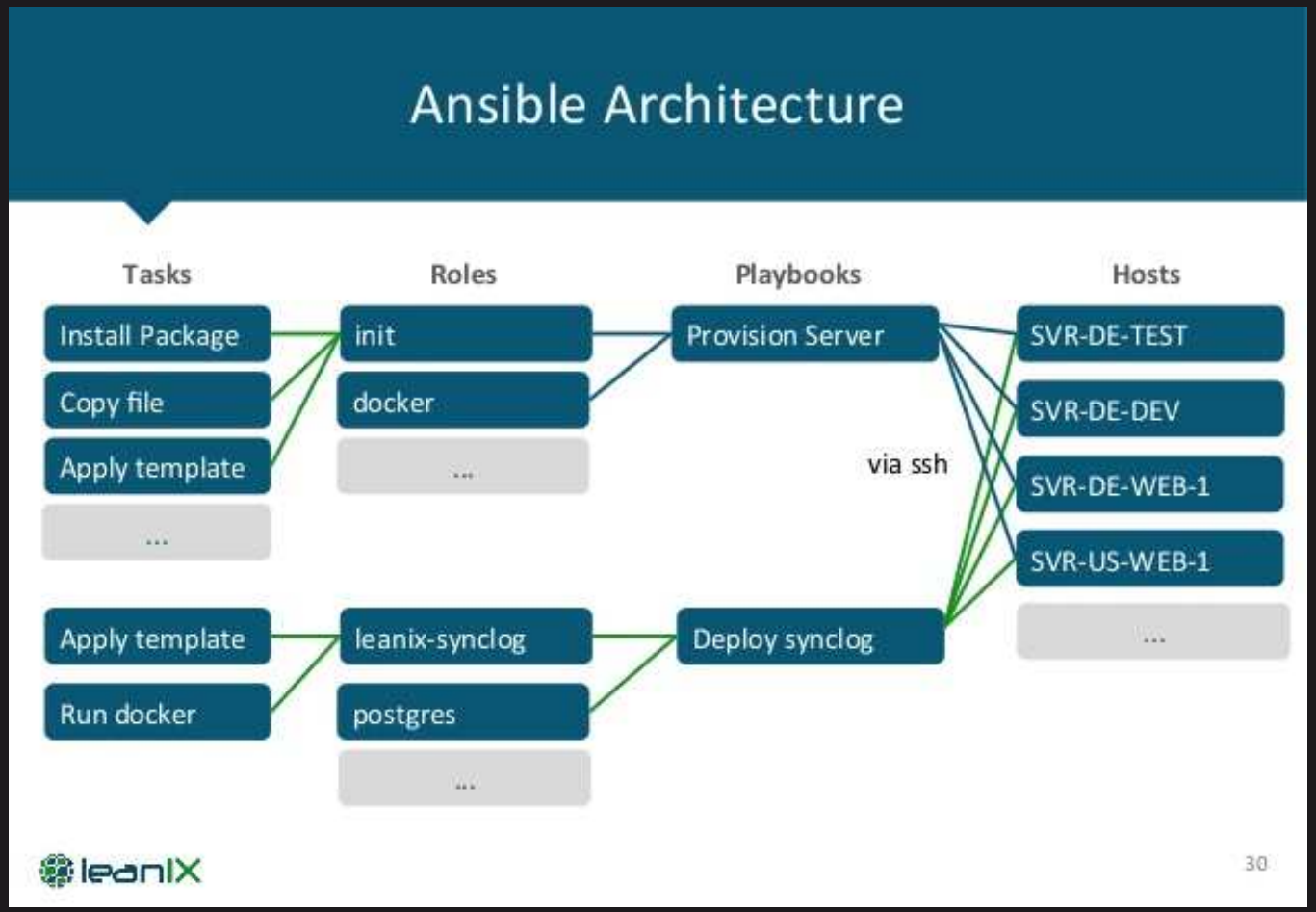
Automatic Providing :

Product or Service or Infrastructure in any Project

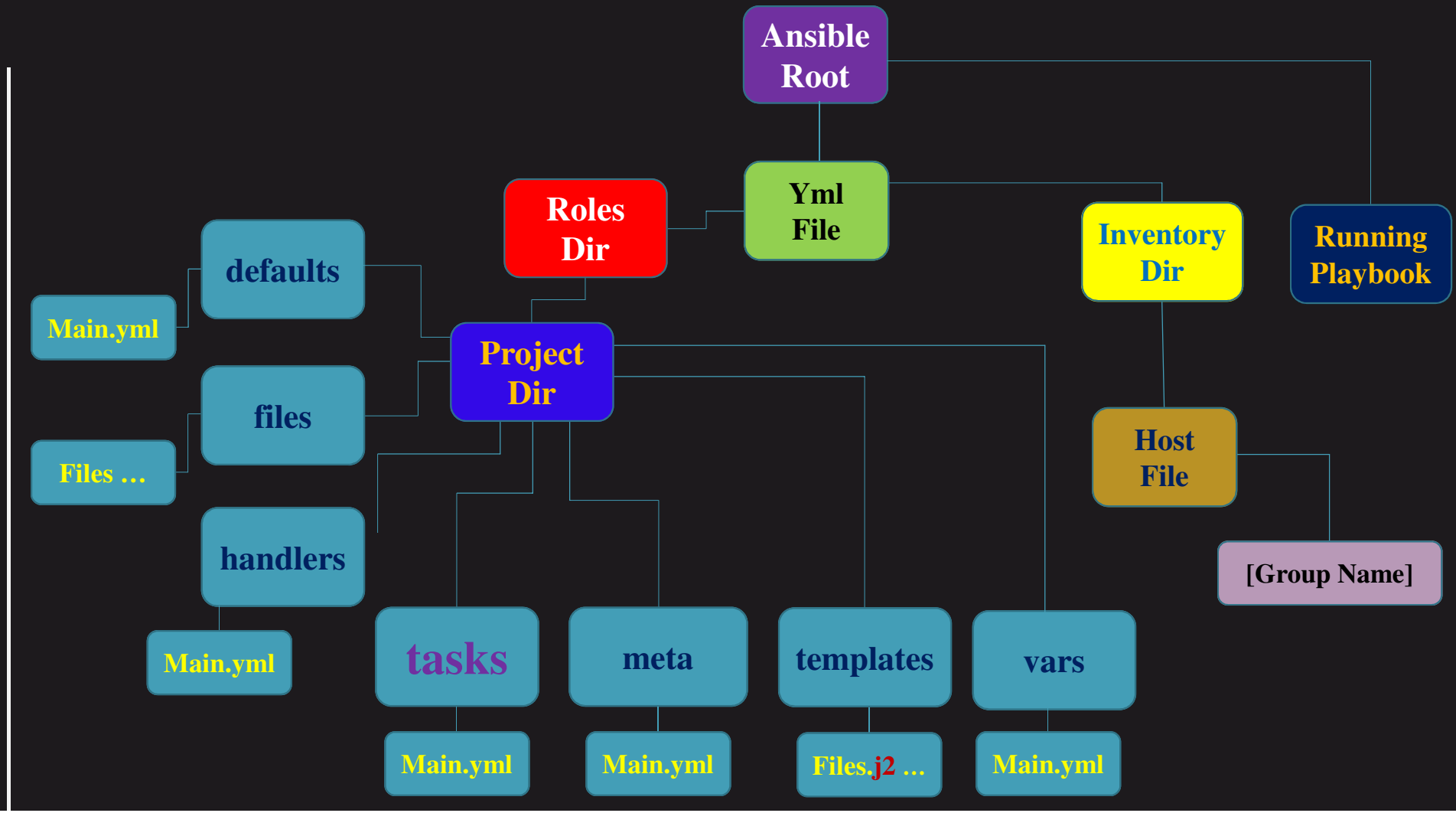
such as :

- Deployment
- Installation
- Change Management

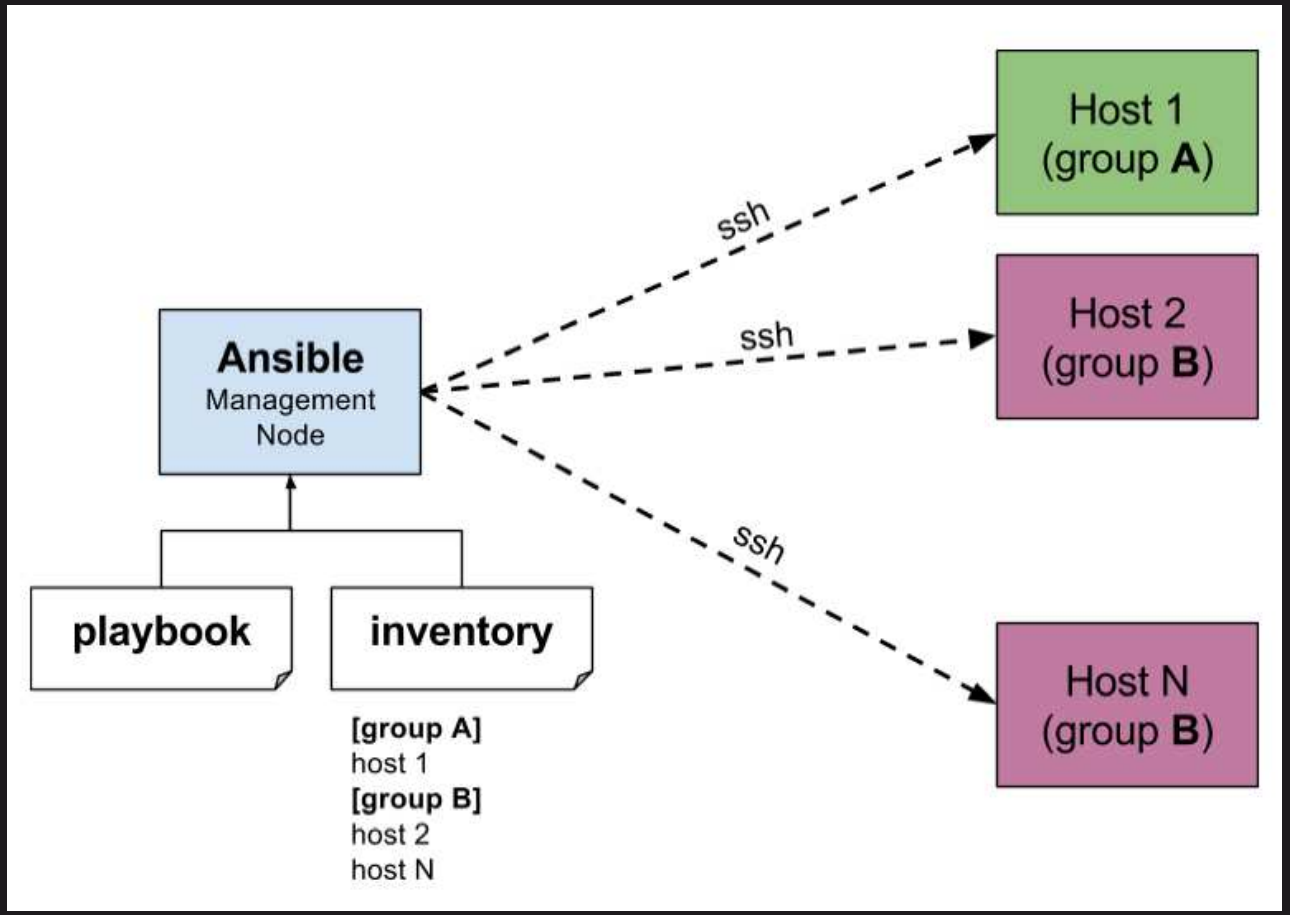
ANSIBLE ARCHITECTURE
AND CORE COMPONENTS OF ANSIBLE



ANSIBLE-PLAYBOOK STRUCTURE



ANSIBLE INVENTORY AND HOSTS



ANSIBLE INVENTORY AND HOSTS

Inventory Host File Sample :

```
[TIDB]
tidb1.lotus.ir
tidb2.lotus.ir

[Live_Switch]
atmlive.lotus.ir
complaintsys1.lotus.ir

[Live-ModernGateways]
mbgw1.lotus.ir
mbgw2.lotus.ir

[Live-ModernBanking_Services]
chmgrwebsrv.lotus.ir
mobilebanking.lotus.ir
Internet-T.lotus.ir
Mobile-T.lotus.ir
Web-T.lotus.ir
```

ANSIBLE RUNNING PLAYBOOK

How to Ansible Running Playbook with Parameters and Error handling in playbooks

ansible-playbook –i inventory/HostsFile_Name FileName.yml --switches

switch :

-v : verbose for view Detail Error (-v , -vv , -vvv)

--tags=tag_name : RUN Only tag_name used into task/main.yml

--tags “tag_name1,tag_name2,…“ : Select tags for RUN

--skip-tags “tag_name,…" : Select tags for Not RUN

--step : (y/n/c) (yes/no/continue)

--list-tasks : List Of Tasks in YML File Without RUN

--extra-vars VarName=“Value” : Create Variable

--syntax-check : yml File Checking Syntax

--check (**Dry-run mode**): Check yml file without RUN and Download

INSTALL AND CONFIGURE ANSIBLE & CREATE THE RSA KEY PAIR

Install and Configure:

- Preinstallation :
  - Setting Hostname
  - Disable Selinux
- yum install epel-release
- yum install ansible

- Vi /etc/ansible/ansible.cfg
  - #sudo_user = root Note : Default
  - #log_path = /var/log/ansible.log Note : Uncomment
  - #deprecation_warnings=False

- Edit /etc/ansible/hosts
  - **[group_name]**
  - **Hostname Short name or Large name**
  - **IP Address**

  Sample :
  - [server] == (Group Name)
    myclient
    myclient.company.com
    192.168.190.156

INSTALL AND CONFIGURE ANSIBLE & CREATE THE RSA KEY PAIR

Create the RSA Key Pair :

The first step is to create the keygen pair on the Server machine and second step Copy to Hosts

- **Create ssh-keygen :**

  ssh-keygen –t rsa

  Once you have entered the Gen Key command, you will get a few more questions:
  Enter file in which to save the key (/home/test/.ssh/id_rsa):
  Enter no password for the next prompt

- **Copy the Public Key**

  ssh-copy-id root@192.168.85.135

  Repeat the same process for other machines you wish to login automatically with.
  Ensure the test username has sudo access to the remote clients

ANSIBLE AD-HOC COMMANDS

Command : TEST Host For Running Ansible

- **ansible -m ping all/GROUP_NAME in /etc/ansible/hosts == server**

OutPut :

```
docker2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

ANSIBLE AD-HOC COMMANDS

An ad-hoc command is :

something that you might type in to do something really quick but don't want to save for later.

For Example :

1. **Parallelism and Shell Commands**
2. **File Transfer**
3. **Managing Packages**
4. **Users and Groups**
5. **Deploying From Source Control**
6. **Managing Services**
7. **Gathering Facts**

ANSIBLE AD-HOC COMMANDS

Note :
vim /etc/hosts :
192.168.190.156 myclient

1. **Parallelism and Shell Commands**

Command :

- ansible -m shell -a '/sbin/reboot' server

OutPut : Reboot Host

Command :

- ansible -m shell -a 'free -m' server

OutPut :

```
docker2 | SUCCESS | rc=0 >>
              total        used        free      shared  buff/cache   available
Mem:            976         321         114          56         541         415
Swap:          1023           1        1022
```

ANSIBLE AD-HOC COMMANDS

2. File Transfer : Copy From Ansible To Host

Command :

- ansible -m copy -a "src=/home/file dest=/opt/file" server

OutPut :

```
192.168.1.110 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": true,
  "checksum": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
  "dest": "/opt/file",
  "gid": 0,
  "group": "root",
  "md5sum": "d41d8cd98f00b204e9800998ecf8427e",
  "mode": "0644",
  "owner": "root",
  "size": 0,
  "src": "/root/.ansible/tmp/ansible-tmp-1568258667.67-144525853706652/source",
  "state": "file",
  "uid": 0
}
```

Note : OUTPUT
Green = No Change
Yellow = Change in host

ANSIBLE AD-HOC COMMANDS

2. File Transfer : Change Mode File on Host

Command :

- ansible -m file -a "dest=/opt/file mode=600" server

OutPut :

```
192.168.1.110 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": true,
  "gid": 0,
  "group": "root",
  "mode": "0600",
  "owner": "root",
  "path": "/opt/myfile.txt",
  "size": 0,
  "state": "file",
  "uid": 0
}

[root@ansible ~]# ssh 192.168.1.110
[root@host ~]# ll /opt/file
-rw------- 1 root root 0 Sep 12 07:59 /opt/file
```

ANSIBLE AD-HOC COMMANDS

3. Managing Packages

Command :

- ansible -m yum -a "name=net-tools state=present" all

OutPut :


```
192.168.1.110 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": true,
  "changes": {
    "installed": [
      "net-tools"
    ]
  },
  "msg": "",
  "rc": 0,
  "results": [
    "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\n * base:
Resolved\n\n================================================================================\n Package
Arch        Version                    Repository
Size\n================================================================================\nInstalling:\n net-tools
x86_64      2.0-0.24.20131004git.el7       base      306 k\n\nTransaction
Summary\n================================================================================\nInstall  1
Package\n\nTotal size: 306 k\nInstalled size: 918 k\nDownloading packages:\nRunning transaction check\nRunning transaction
test\nTransaction test succeeded\nRunning transaction\n  Installing : net-tools-2.0-0.24.20131004git.el7.x86_64                1/1 \n  Verifying
: net-tools-2.0-0.24.20131004git.el7.x86_64                1/1 \n\nInstalled:\n  net-tools.x86_64 0:2.0-0.24.20131004git.el7
\n\nComplete!\n"
  ]
}
```

ANSIBLE AD-HOC COMMANDS

4. Users and Groups

Command :

- ansible -m user -a "name=usertest state=present" all

OutPut :

```
192.168.1.110 | CHANGED => {
   "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python"
   },
   "changed": true,
   "comment": "",
   "create_home": true,
   "group": 1001,
   "home": "/home/usertest",
   "name": "usertest",
   "shell": "/bin/bash",
   "state": "present",
   "system": false,
   "uid": 1001
}

[root@ansible ~]# ssh 192.168.1.110
[root@host ~]# cat /etc/passwd | grep usertest
usertest:x:1001:1001::/home/usertest:/bin/bash
```

ANSIBLE AD-HOC COMMANDS

5. Deploying From Source Control

Command :

```
ansible -m git -a "repo=https://foo.example.org/repo.git  dest=/srv/myapp "  server
```

ANSIBLE AD-HOC COMMANDS

6. Managing Services

Command :

```
ansible   -m  service   -a "name=httpd    state=restarted " server
```

ANSIBLE AD-HOC COMMANDS

7. Gathering Facts

gather_facts Module :

This module takes care of executing the configured facts modules
The default is to use the setup module.

Command : gather_facts or setup

- ansible -m setup server OR ansible -m gather_facts server

OutPut :

server All Info

ANSIBLE VARIABLE

Define and Call and Used Place Variables

Where :

- Variables Defined in a Playbook
- Using Variable :About Jinja2
- Variables Defined in a vars
- Variables Defined in a default
- Variables Defined in a task

Create :

```
vars:
  Name: blue
```

Call :

```
{{ Name }}
```

ANSIBLE VARIABLE

INVALID Variable Names :

- mysql version (multiple words)
- mysql.port (a dot)
- 5 (a number)
- mysql-port (a dash)

Example :

foo_port is a great variable. foo5 is fine too.

foo-port, foo port, foo.port and 12 are not valid variable names.

Reserved Keys : Do not use as a variable

add, append, as_integer_ratio, bit_length, capitalize, center, clear, conjugate, copy, count, decode, denominator, difference, difference_update, discard, encode, endswith, expandtabs, extend, find, format, fromhex, fromkeys, get, has_key, hex, imag, index, insert, intersection, intersection_update, isalnum, isalpha, isdecimal, isdigit, isdisjoint, is_integer, islower, isnumeric, isspace, issubset, issuperset, istitle, isupper, items, iteritems, iterkeys, itervalues, join, keys, ljust, lower, lstrip, numerator, partition, pop, popitem, real, remove, replace, reverse, rfind, rindex, rjust, rpartition, rsplit, rstrip, setdefault, sort, split, splitlines, startswith, strip, swapcase, symmetric_difference, symmetric_difference_update, title, translate, union, update, upper, values, viewitems, viewkeys, viewvalues, zfill.

ANSIBLE VARIABLE

Define Cascading (hierarchy) variable and call it

1 – vars/main.yml

```yaml
lotus:
 env:
  version: '1.3.3.0'
  name:  'TEST'
 deployment:
  url: 'http://centdns1.lotus.ir/deployment'
  username: 'coreuser'
  password: '123456789'
 override:
  jms:
   url: 't3://localhost:7001'
```

2 – tasks/main.yml

```yaml
 - name: Download lotus core jar file
  get_url: url={{ lotus.deployment.url }}/main-with-all-dependencies.jar url_username={{ lotus.deployment.username }}
 url_password={{ lotus.deployment.password }} dest=/var/lotus/libs/main-with-all-dependencies-{{ lotus.env.name }}.jar
 mode=0444
  notify: restart lotus core
  tags: [ 'lotus_core' ]
```

ANSIBLE MANAGING
A TASK, HANDLERS, AND TAGS IN BLOCK PLAYBOOK

Introduce Ansible Block :

- **Task and Handlers and any main.yml files**
- **Tags and Name and any Modules**

```
- name: Block Description
  Module_Type: Module Command Structure Part1=value1  and   Part2=value2  and …
  tags: [TagName_Block_Description]

OR

- name: Block Description
  Module_Type:
        Module Command Structure Part1: value1
        Module Command Structure Part1: value2
        …
  tags: [TagName_Block_Description]
```

CREATE ANSIBLE-PLAYBOOK STRUCTURE FOR GETTING SETUP

Create Main Structure :

1- mkdir /home/ansible/provision -p
2- vim /home/ansible/provision/ProjectName.yml
3- mkdir /home/ansible/provision/inventory
4- mkdir /home/ansible/provision/roles

```
[root@ansible ansible]# pwd
/home/ansible
[root@ansible ansible]# tree
.
└── provision
    ├── ProjectName.yml
    ├── inventory
    └── roles

3 directories, 1 file
```

CREATE ANSIBLE-PLAYBOOK STRUCTURE FOR GETTING SETUP

Main YML FILE :

vim /home/ansible/provision/ProjectName.yml

vim ProjectName.yml == (Example install_nginx.yml)

```
---
- hosts: HostsFile_Name  == (File of  list Servers And IPs  into Inventory Directory)
  roles:                 == (roles Directory )
  - ProjectName   == (Directory Project name in roles == Example nginx)
```

Inventory :

vim /home/ansible/provision/inventory/HostsFile_Name

[HostsFile_Name] ➔ (Server Group_Name)

IP_Address1 OR DNS_Names1 ➔ Sample : 192.168.102.170 OR test.test.com

IP_Address2 OR DNS_Names2

...

CREATE ANSIBLE-PLAYBOOK STRUCTURE FOR GETTING SETUP

Roles :

cd /home/ansible/provision/roles/**ProjectName**

mkdir

- ✓ tasks = main.yml
- ✓ defaults = main.yml
- ✓ vars = main.yml
- ✓ files = Source files (nginx.tar.gz)
- ✓ templates = config files with jinja2 extenstion (httpd.conf.j2)
- ✓ handlers = main.yml
- ✓ meta = main.yml

RUN :

ansible-playbook -i inventory/HostsFile_Name ProjectName.yml

CREATE ANSIBLE-PLAYBOOK STRUCTURE OVERVIEW

OverView Structure :

vim /home/ansible/provision/myproject.yml

```
---
- hosts: myhosts
  roles:
   - myproject
```

vim /home/ansible/provision/inventory/myhosts

```
[myhosts]
192.168.102.102
[....]
.....
```

```
mkdir -p /home/ansible/provision/roles/myproject/defaults
mkdir -p /home/ansible/provision/roles/myproject/files
mkdir -p /home/ansible/provision/roles/myproject/handlers
mkdir -p /home/ansible/provision/roles/myproject/meta
mkdir -p /home/ansible/provision/roles/myproject/tasks
mkdir -p /home/ansible/provision/roles/myproject/templates
mkdir -p /home/ansible/provision/roles/myproject/vars
```

CREATE ANSIBLE-PLAYBOOK STRUCTURE OVERVIEW

- **On Host : (IP : 192.168.190.156)**
  - **1- FireWall :**
    - ✓ **iptables stop**
      Or
    - ✓ **create Accept rule for ansible server (IP and port 22)**
  - **2- Set Hostname :**
    - ✓ **hostnamectl set-hostname myhost**
- **On Ansible Server : (IP : 192.168.190.155)**
  - **1- Create ssh Keygen and ssh Copy-ID :**
    - ✓ **ssh-keygen -t rsa**
    - ✓ **ssh-copy-id 192.168.190.156**
    - ✓ **For Test : ssh 192.168.190.156 Connect without Password**
    - ✓ **Ping Pong Test**
  - **2- Create Main Structure :**
    - ✓ vim /home/ansible/provision/myproject.yml =====> hosts: myhosts
    - ✓ vim /home/ansible/provision/inventory/myhost ==== > [myhosts]
    - ✓ mkdir /home/ansible/provision/roles/myproject
      - defaults files handlers meta tasks templates vars
  - **3- Write Main.YML and Files o Templates on roles Directory**

ANSIBLE LABS AND USED MODULES

Lab1 - Create Directory :

vim /home/ansible/provision/roles/myproject/tasks/main.yml

```
- name: create   directory
  file: path=/home/testdir1 state=directory
  tags: [createdir]

OR

- name: create   directory
  file:
    path:  /home/testdir2
    state:   directory
  tags: [createdir]
```

RUN : ansible-playbook -i inventory/myhost myproject.yml

ANSIBLE LABS AND USED MODULES

Lab2 - Create Directory With Owner and Mode :

vim /home/ansible/provision/roles/myproject/tasks/main.yml

```
- name: create   directory
  file: path=/home/testdir1 state=directory  owner=root  group=root  mode=0775    OR   "u=rw,g=r,o=r"
  tags: [createdir]

OR

- name: create   directory
  file:
    path:  /home/testdir2
    state:   directory
    owner:  root
    group:  root
    mode:  0775   OR  "u=rw,g=r,o=r"
  tags: [createdir]
```

Note1 : mode: 0755
mode : "u=rw,g=r,o=r"

Note2 : User=root (User Should be defined)

RUN : ansible-playbook -i inventory/myhost myproject.yml