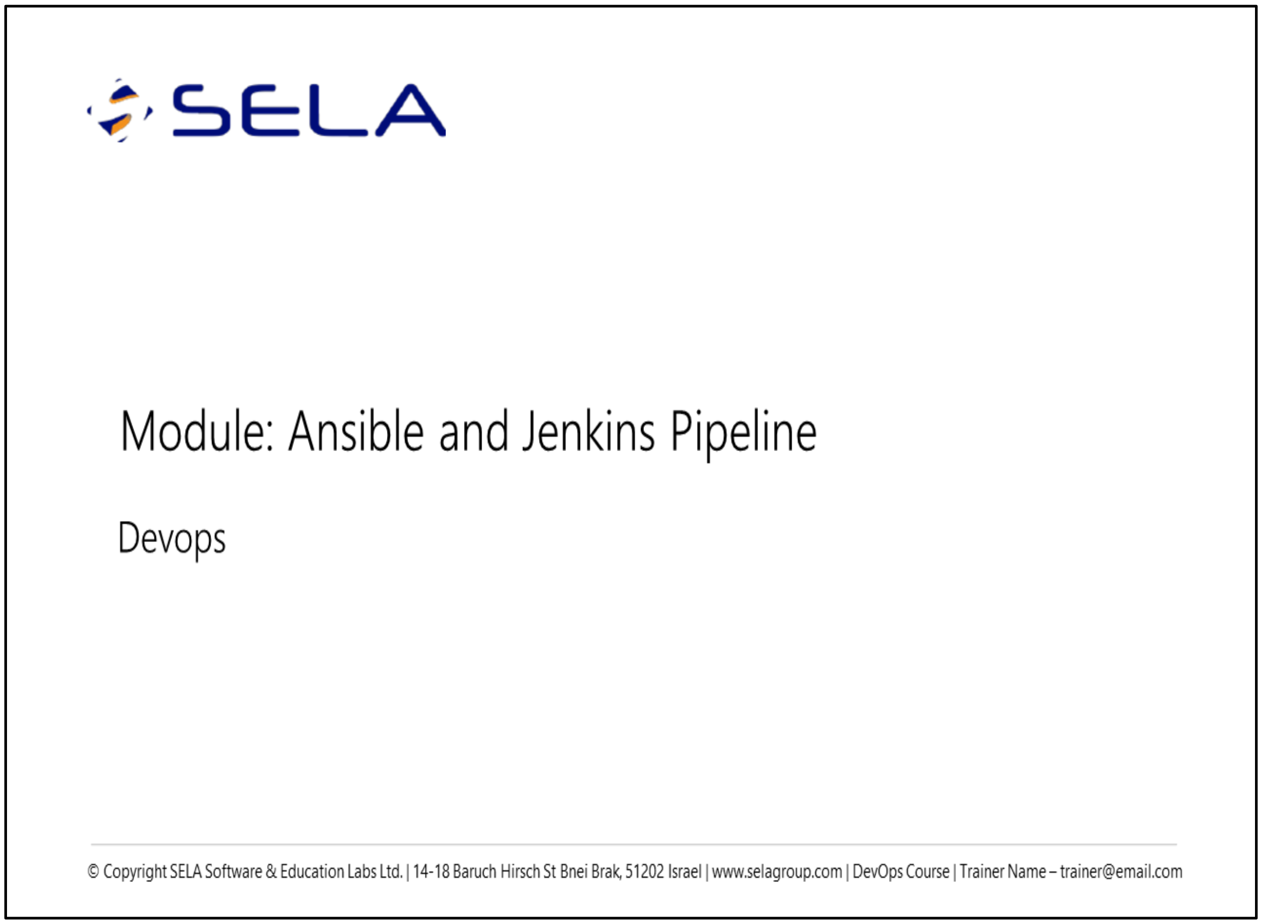
# Ansible and Jenkis Pipeline

DevOps ®

Version 1.0

Sela College

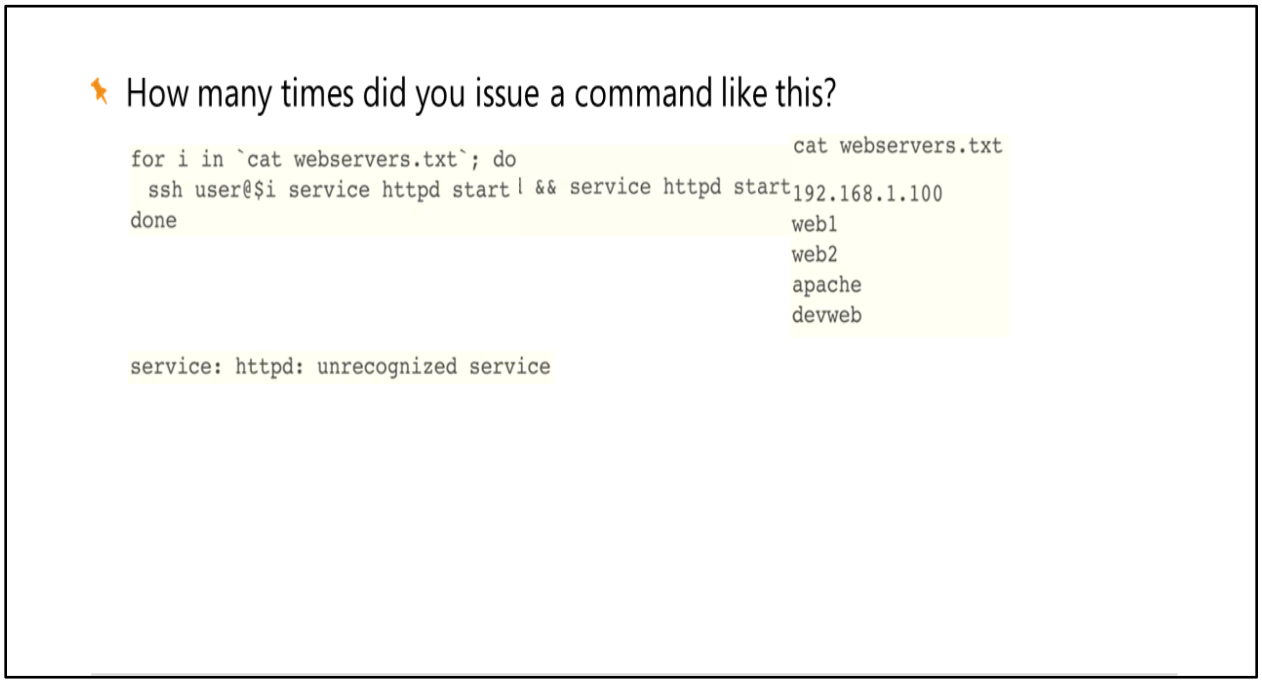
## Table of Contents



## Module: Ansible and Jenkins Pipeline

Devops

## Why Ansible?



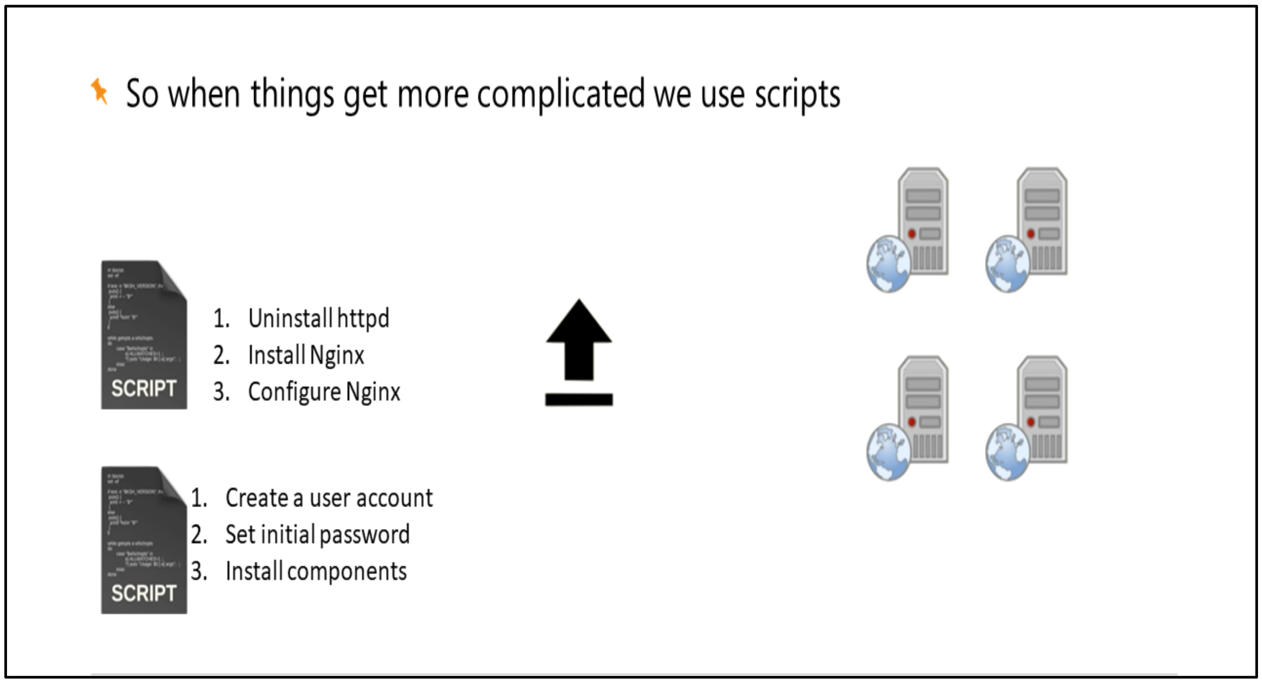
How many times did you issue a command like this?

```
for i in `cat webservers.txt`; do
  ssh user@$i service httpd start | && service httpd start
done
```

```
cat webservers.txt
192.168.1.100
web1
web2
apache
devweb
```

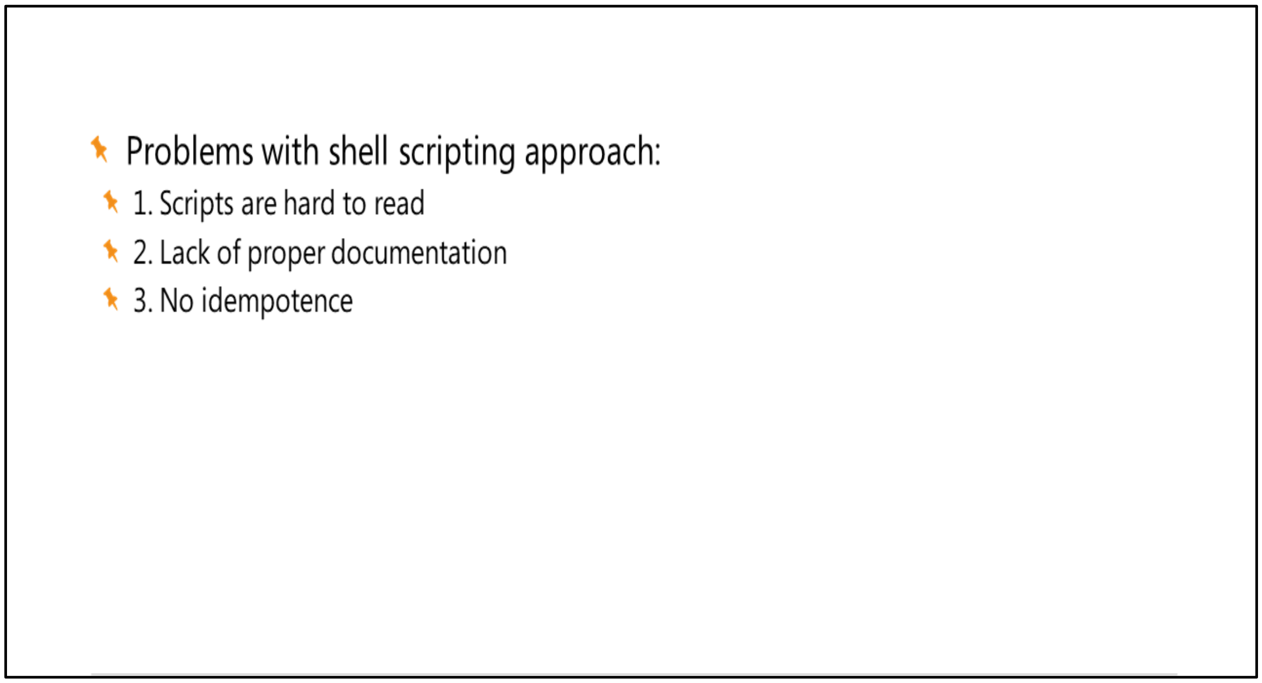
```
service: httpd: unrecognized service
```

## Why Ansible?



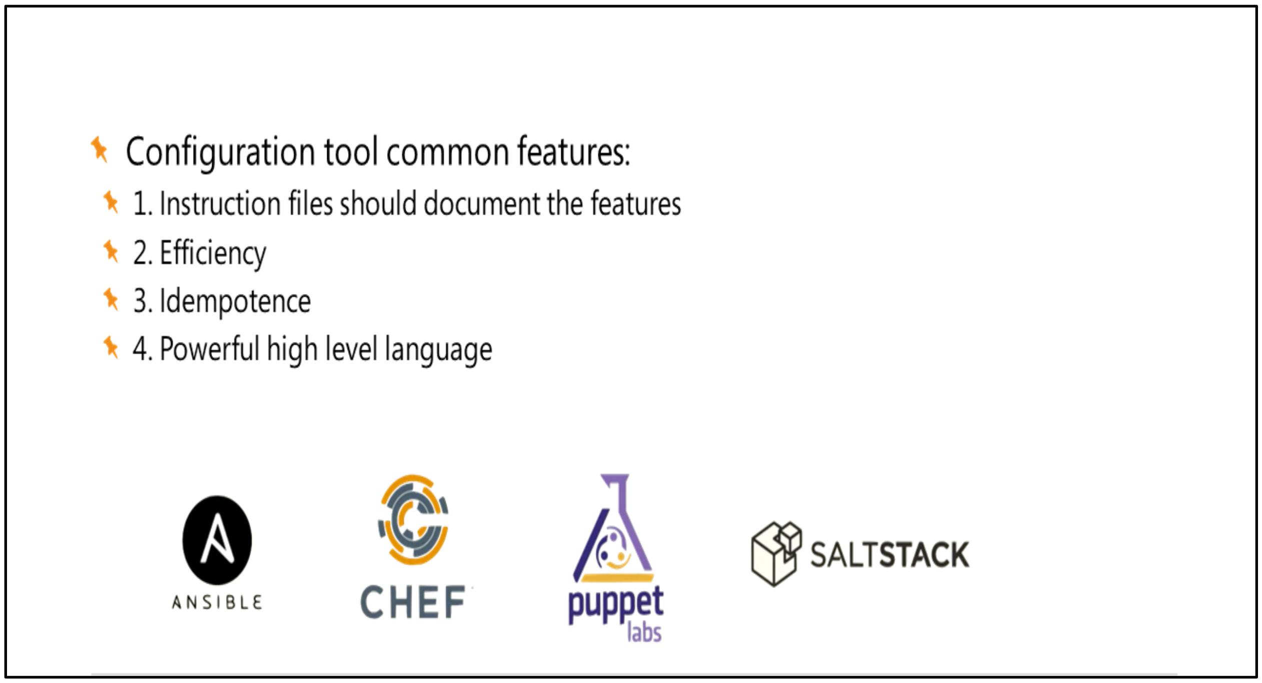
So when things get more complicated we use scripts

## Why Ansible?



- Problems with shell scripting approach:
  - 1. Scripts are hard to read
  - 2. Lack of proper documentation
  - 3. No idempotence

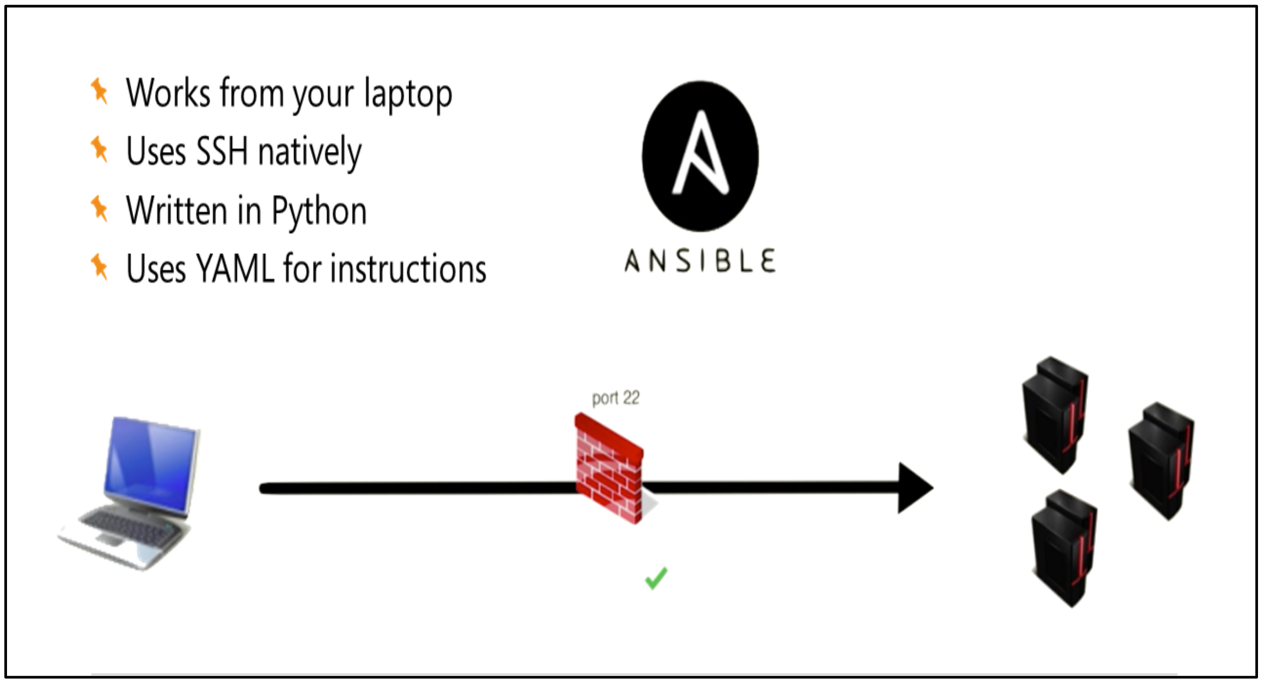
## Why Ansible?



- Configuration tool common features:
  - 1. Instruction files should document the features
  - 2. Efficiency
  - 3. Idempotence
  - 4. Powerful high level language

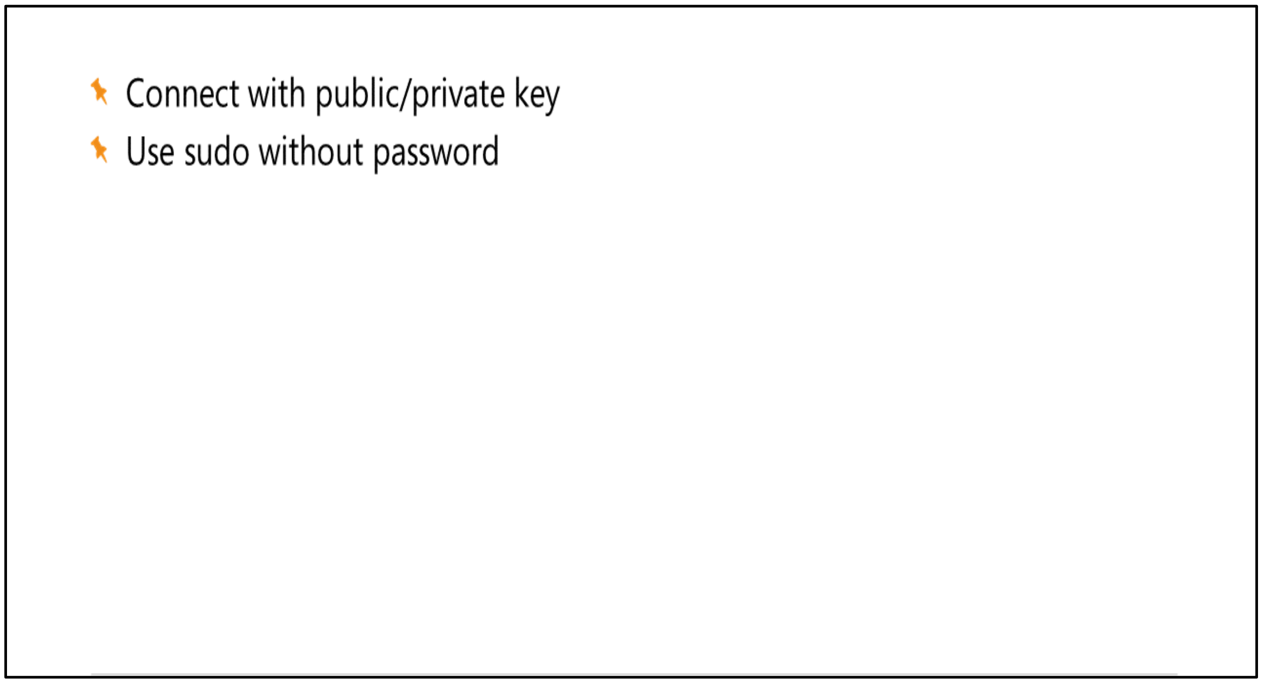
ANSIBLE
CHEF
puppet labs
SALTSTACK

## Why Ansible?



- Works from your laptop
- Uses SSH natively
- Written in Python
- Uses YAML for instructions

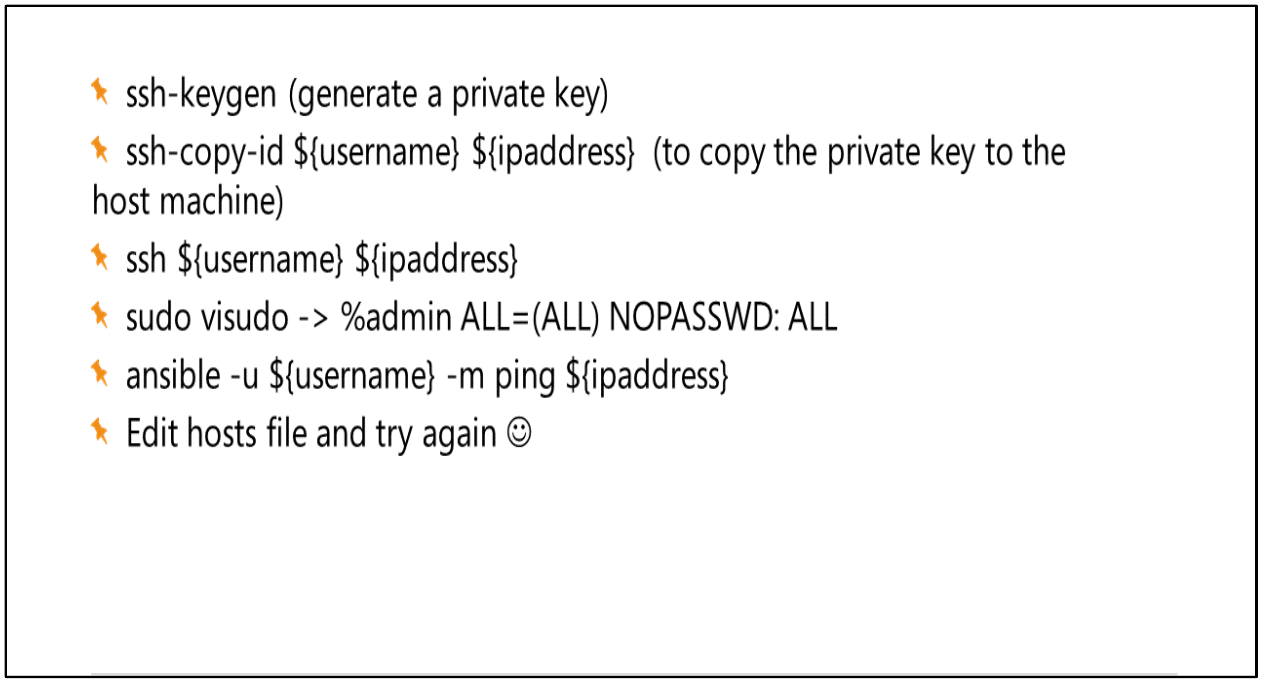
## Prerequisites



- Connect with public/private key
- Use sudo without password

## Demo 1: First Ansible Task

## Demo1 - First Ansible Task

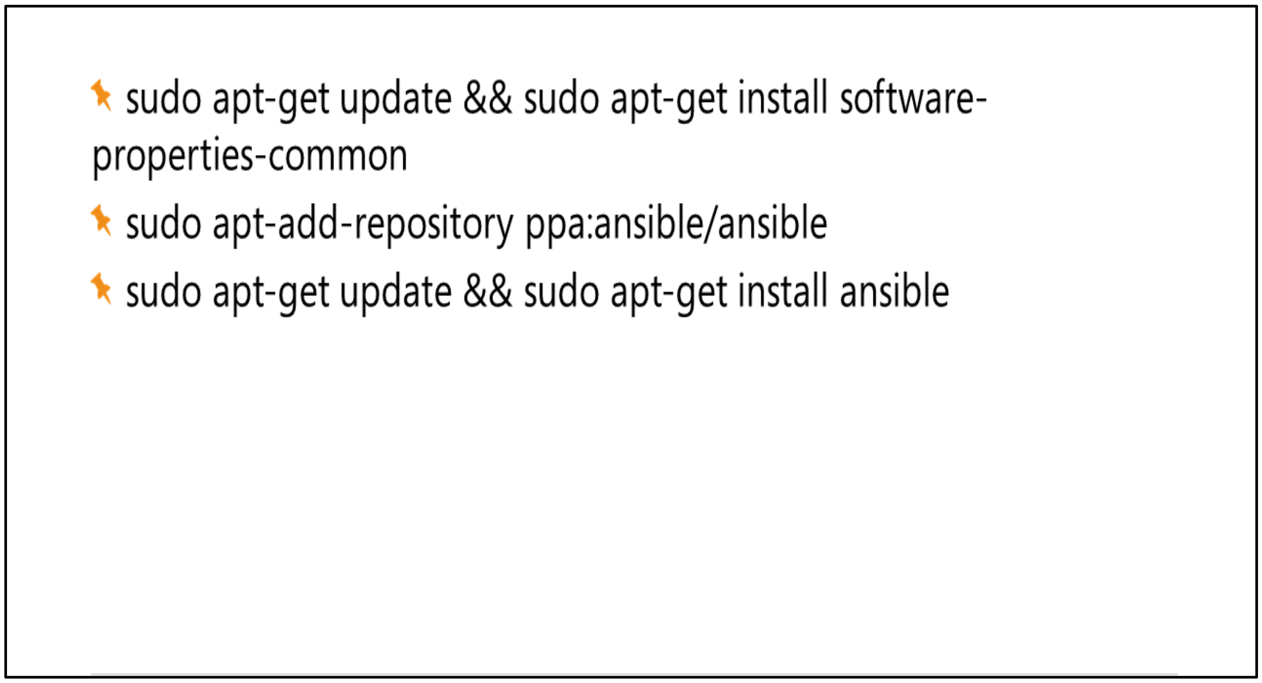


- ssh-keygen (generate a private key)
- ssh-copy-id ${username} ${ipaddress} (to copy the private key to the host machine)
- ssh ${username} ${ipaddress}
- sudo visudo -> %admin ALL=(ALL) NOPASSWD: ALL
- ansible -u ${username} -m ping ${ipaddress}
- Edit hosts file and try again ☺

## Lab: Lab 1: Install Ansible

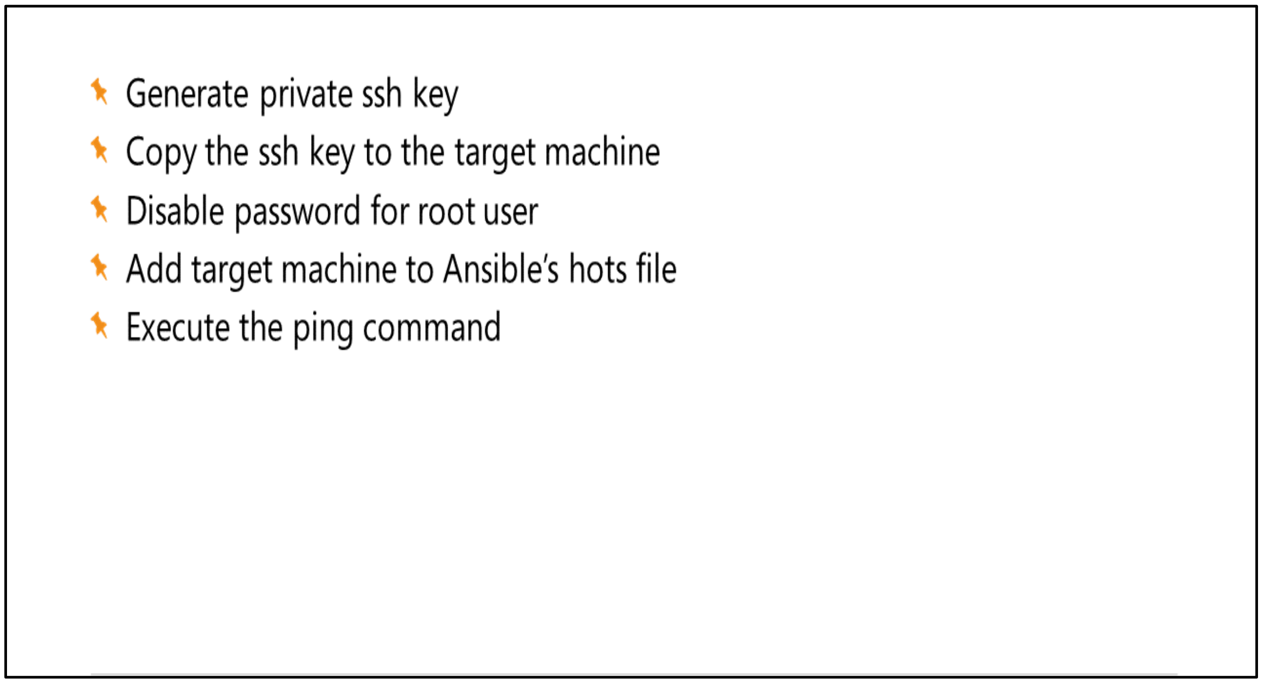
Lab

## Lab 1: Install Ansible


- sudo apt-get update && sudo apt-get install software-properties-common
- sudo apt-add-repository ppa:ansible/ansible
- sudo apt-get update && sudo apt-get install ansible

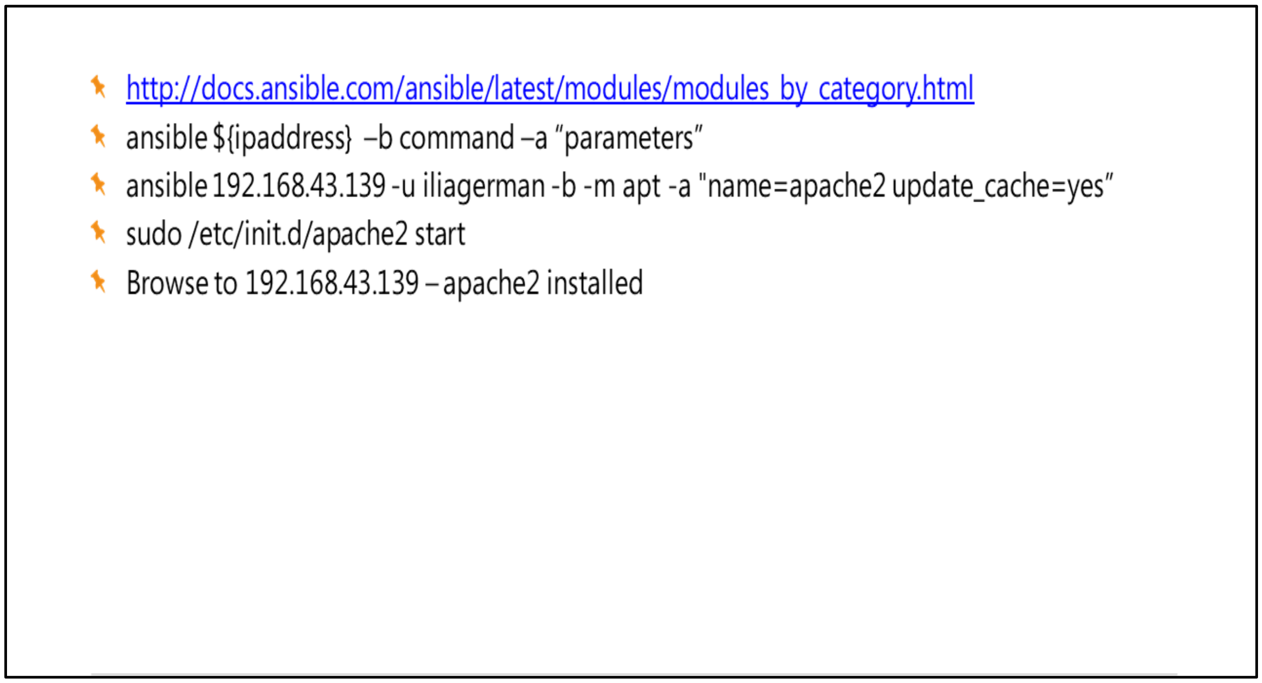
## Lab: Lab 2: Ansible Hello World

## Lab 2: Ansible Hello World



- Generate private ssh key
- Copy the ssh key to the target machine
- Disable password for root user
- Add target machine to Ansible's hots file
- Execute the ping command

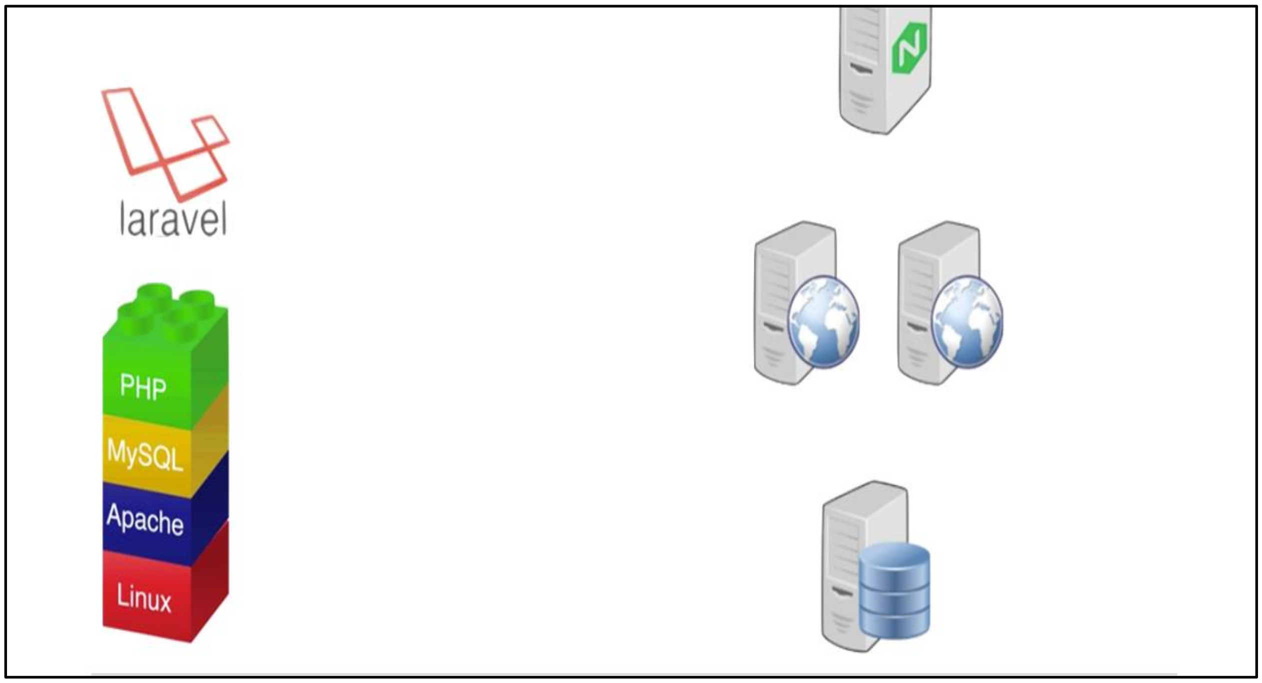
## Executing Ansible Arbitrary Tasks



- http://docs.ansible.com/ansible/latest/modules/modules_by_category.html
- ansible ${ipaddress} –b command –a "parameters"
- ansible 192.168.43.139 -u iliagerman -b -m apt -a "name=apache2 update_cache=yes"
- sudo /etc/init.d/apache2 start
- Browse to 192.168.43.139 – apache2 installed

## Lab: Lab 3: Install Nginx

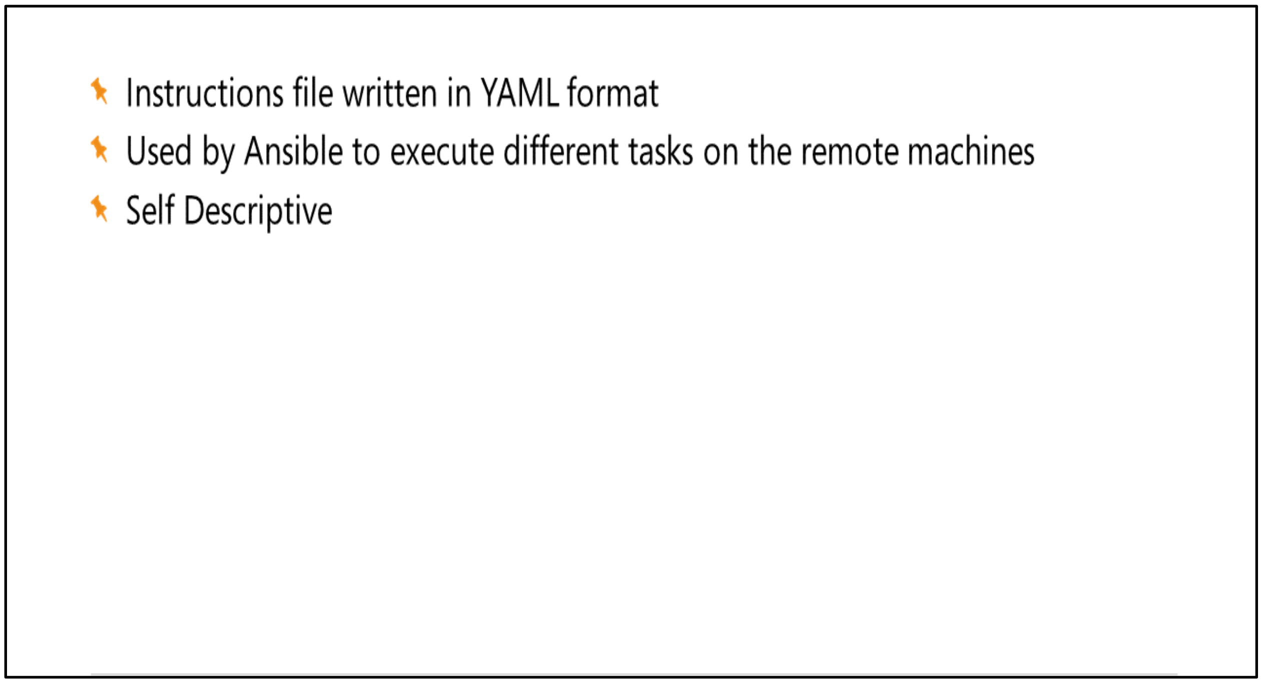
## LAMP Stack



## Manage hosts file

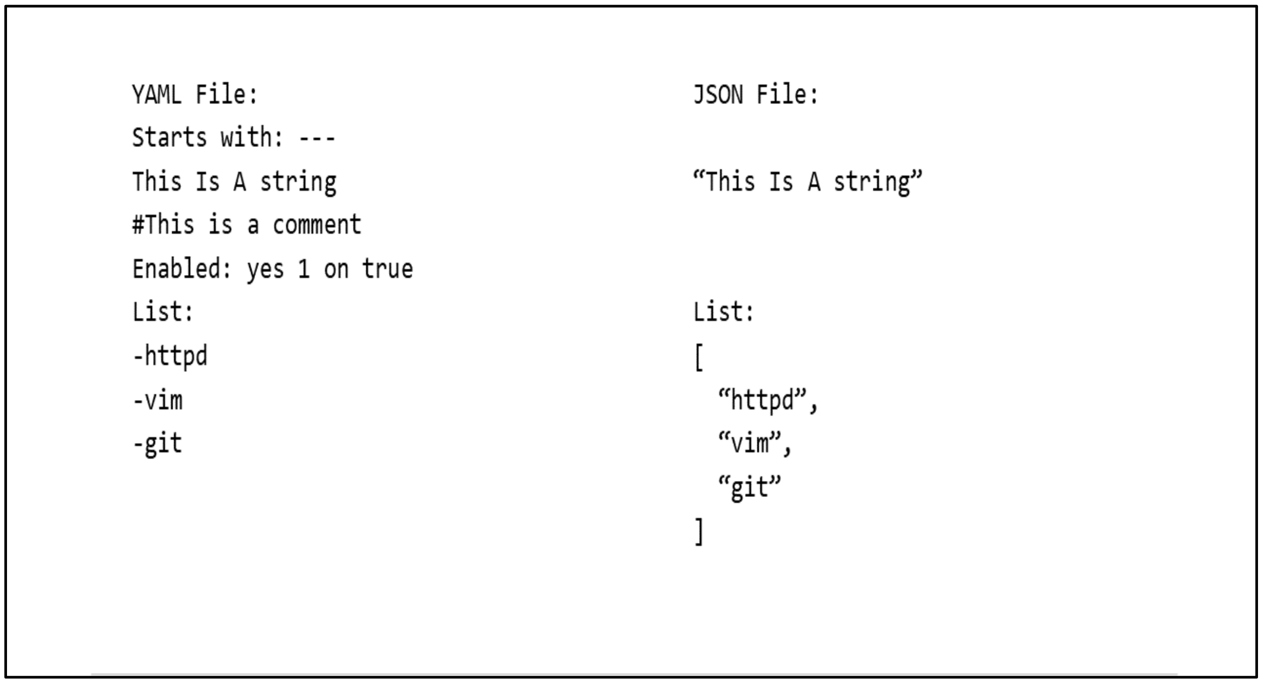
- Add the environment to hosts file
- Modify Ansible hosts file

## Playbooks



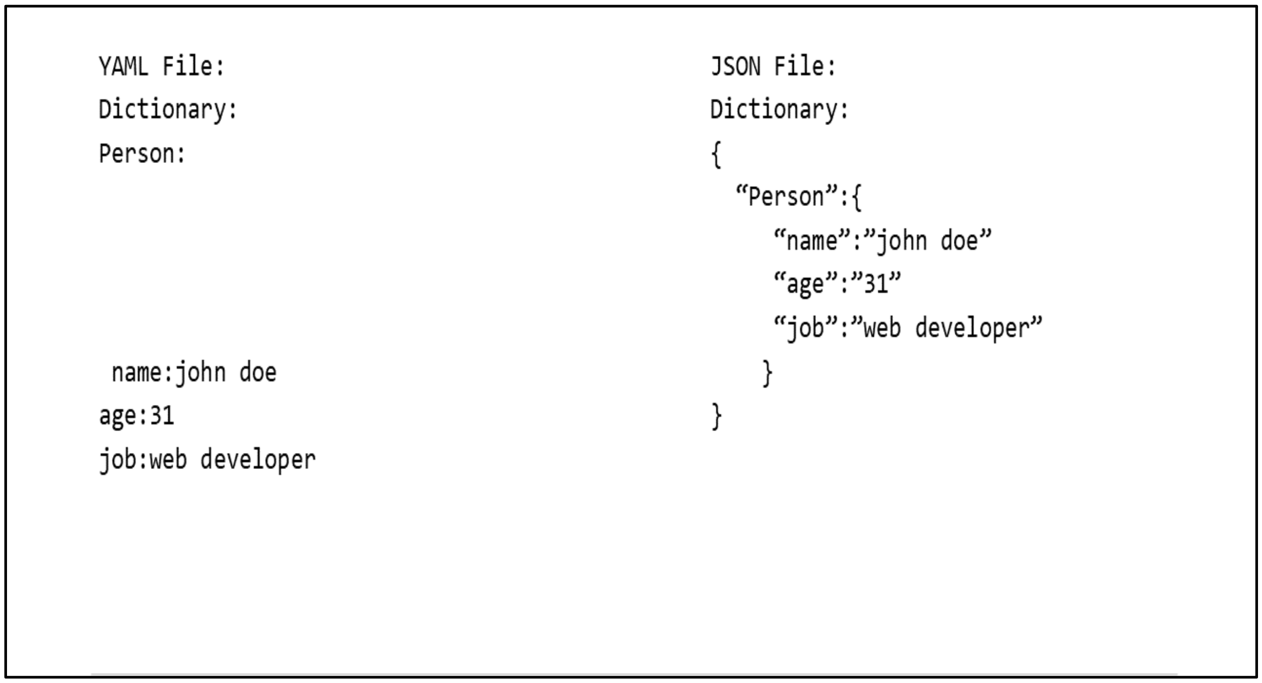
- Instructions file written in YAML format
- Used by Ansible to execute different tasks on the remote machines
- Self Descriptive

## YAML



```
YAML File:                                  JSON File:
Starts with: ---
This Is A string                            “This Is A string”
#This is a comment
Enabled: yes 1 on true
List:                                       List:
-httpd                                      [
-vim                                          “httpd”,
-git                                          “vim”,
                                              “git”
                                            ]
```

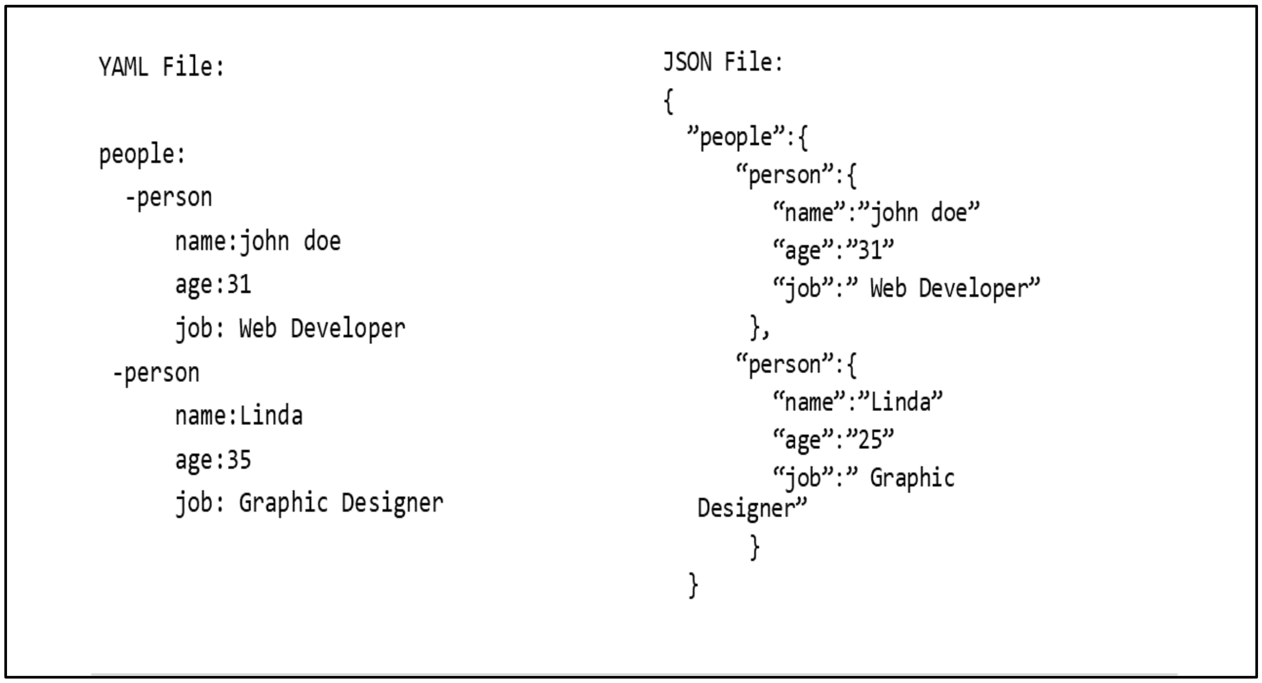
## YAML



```
YAML File:
Dictionary:
Person:




 name:john doe
age:31
job:web developer
```

```
JSON File:
Dictionary:
{
  “Person”:{
     “name”:”john doe”
     “age”:”31”
     “job”:”web developer”
    }
}
```

## YAML



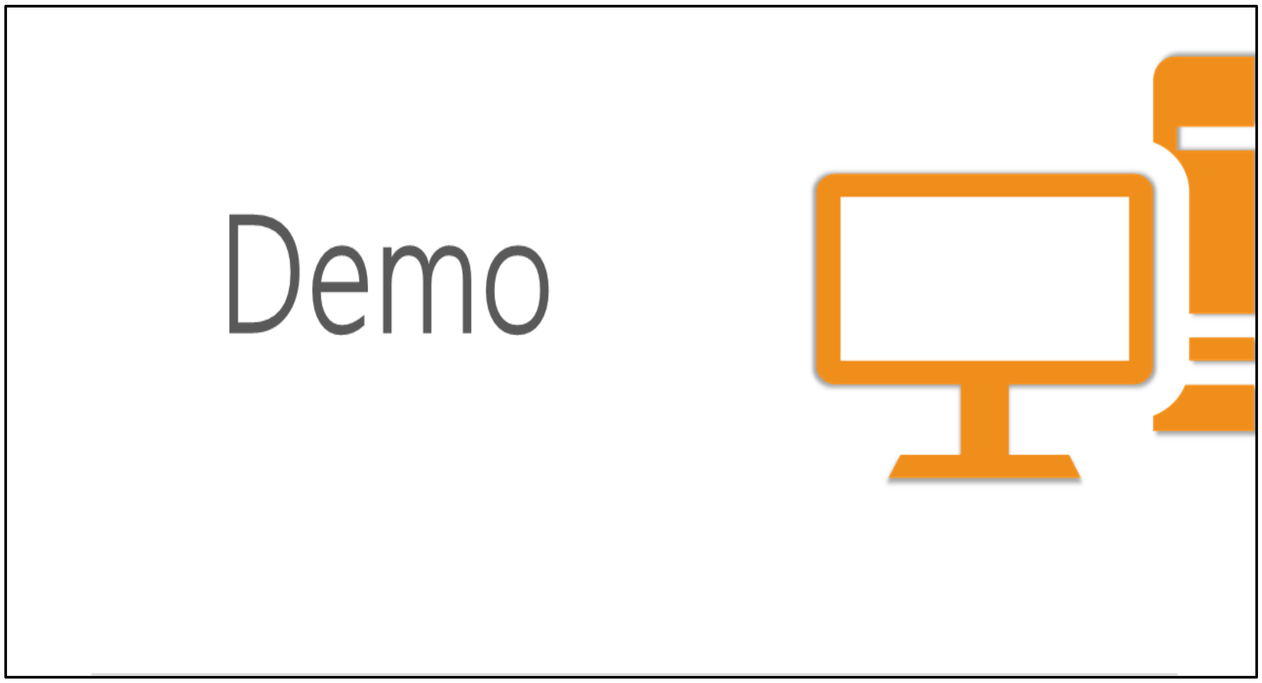
```
YAML File:

people:
  -person
      name:john doe
      age:31
      job: Web Developer
 -person
      name:Linda
      age:35
      job: Graphic Designer
```

```
JSON File:
{
  ”people”:{
      “person”:{
         “name”:”john doe”
         “age”:”31”
         “job”:” Web Developer”
        },
      “person”:{
         “name”:”Linda”
         “age”:”25”
         “job”:” Graphic
   Designer”
        }
  }
```

## Demo2: Install Apache2 Playbook

Demo

## Demo 3: Install PHP

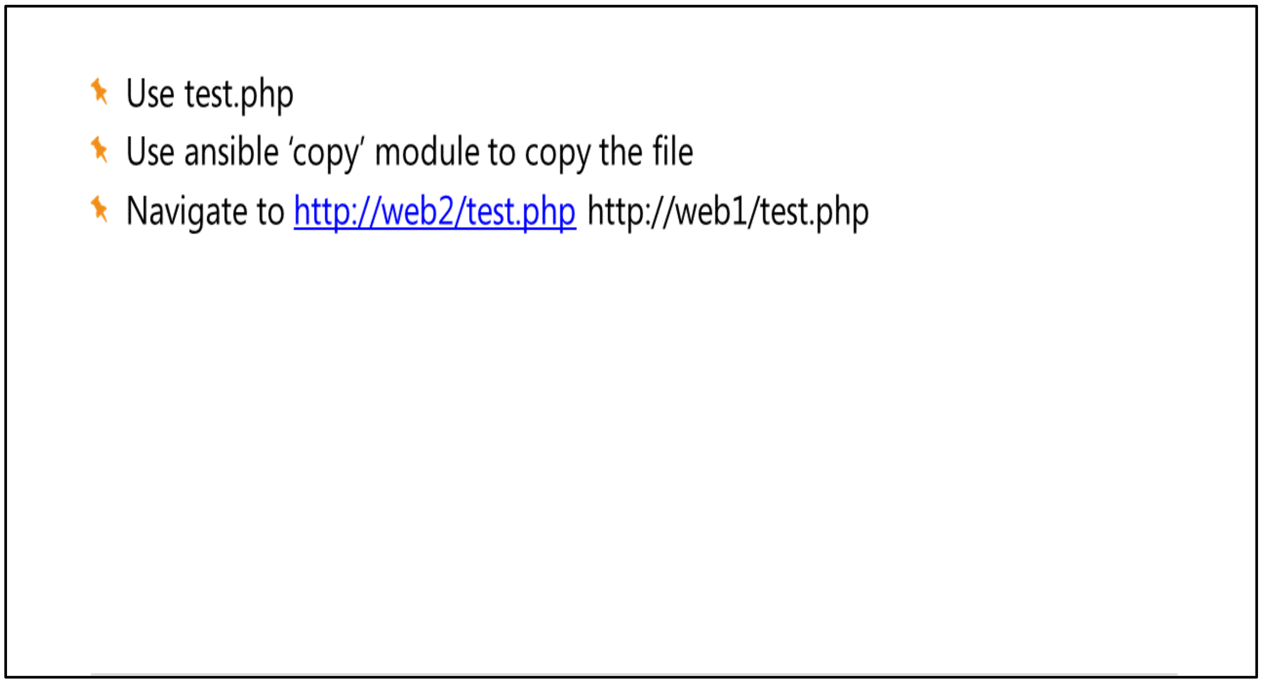


Demo

## Lab: Lab 3: Test PHP Installation

Lab

## Lab 3: Test PHP Installation



- Use test.php
- Use ansible 'copy' module to copy the file
- Navigate to http://web2/test.php http://web1/test.php

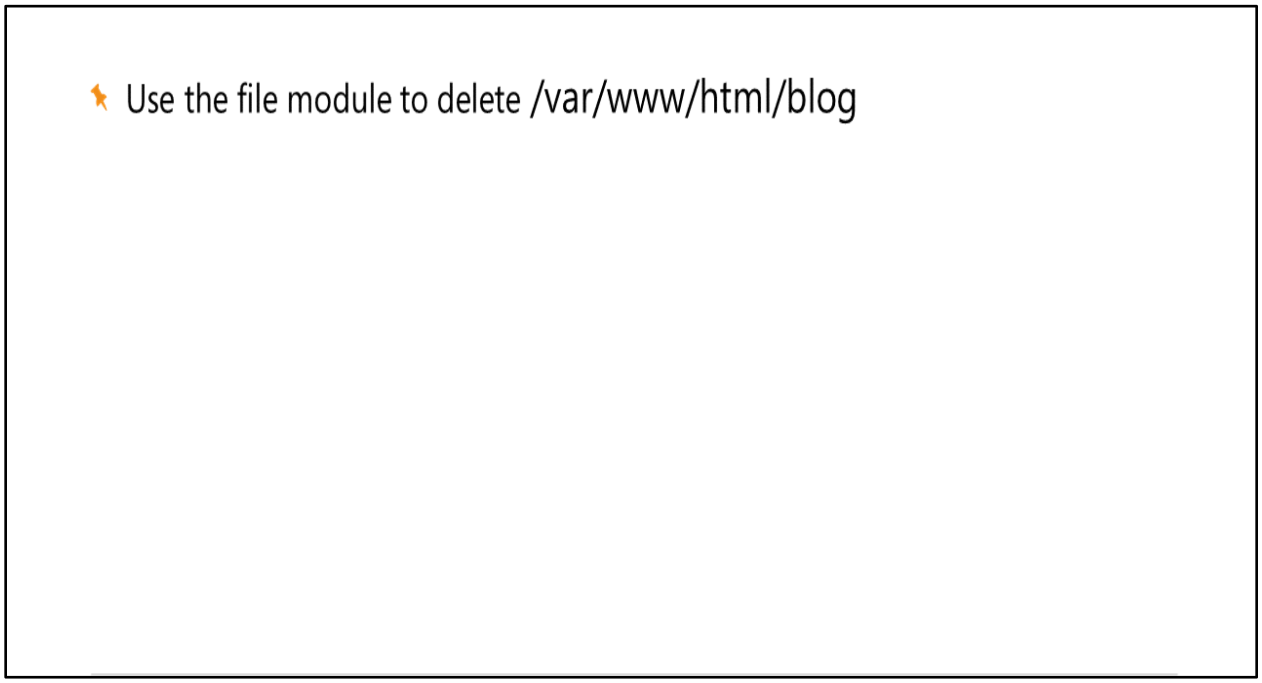
## Demo4: Install Composer + Laravel

# Demo

## Lab: Lab 4: Fix Laravel Installation

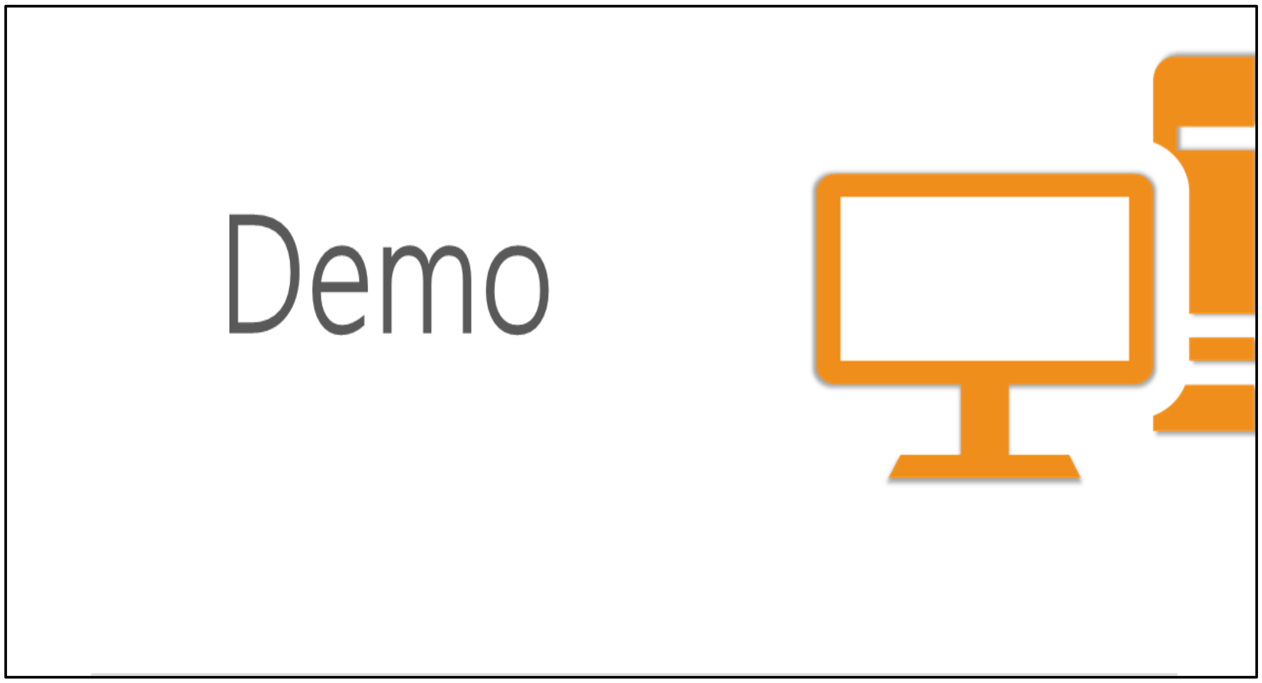
# Lab

## Fix Laravel Installation



- Use the file module to delete /var/www/html/blog

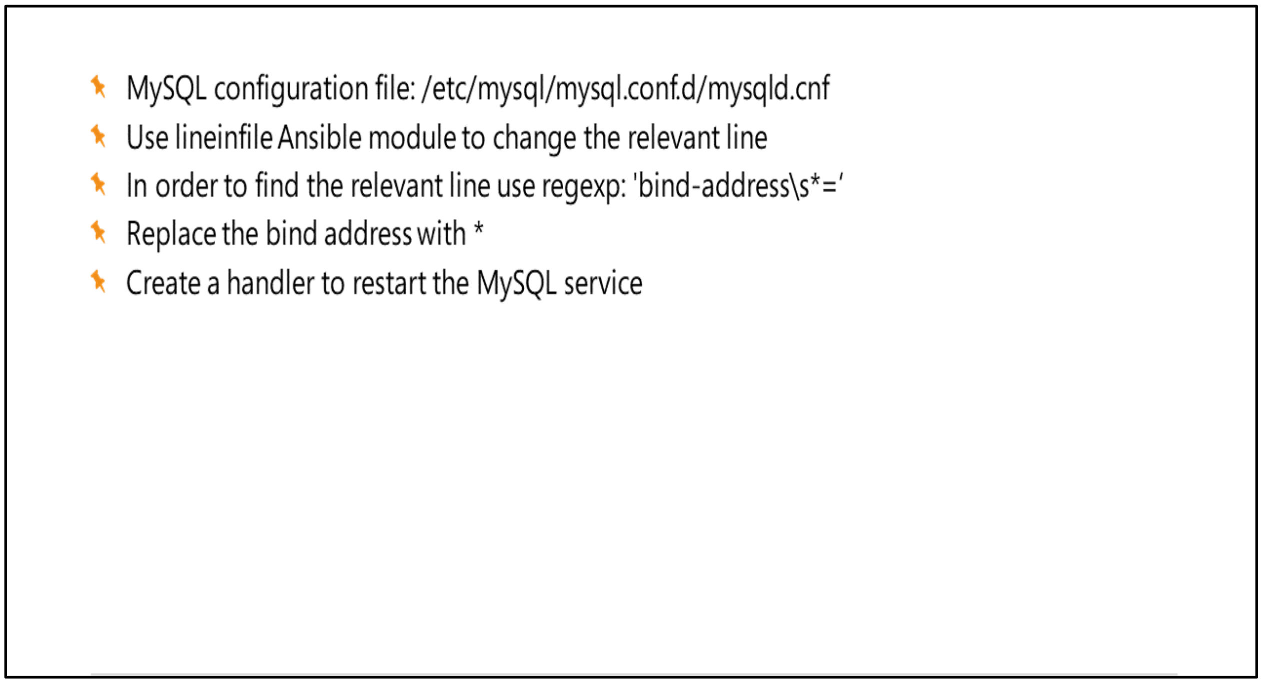
## Demo5: Install MySQL



## Demo6: Configure MySQL Security

## Lab: Lab 5: Fix MySQL Bind-Address

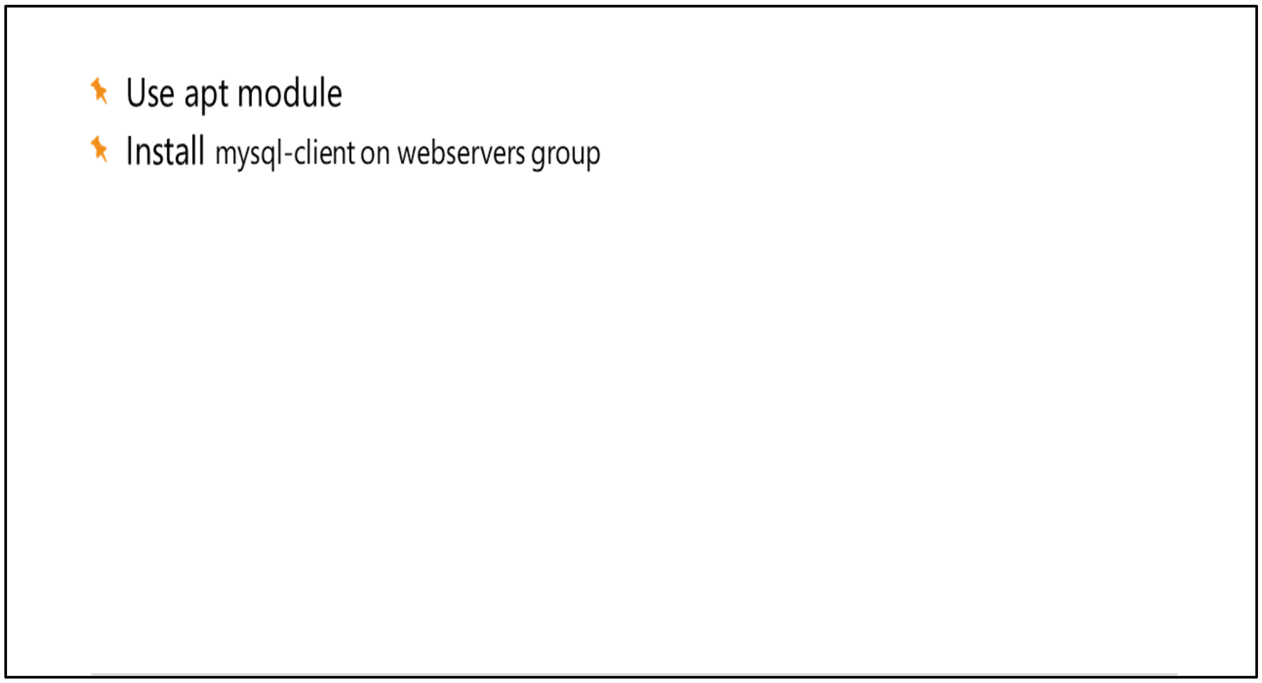
## Fix MySQL Bind-Address



- MySQL configuration file: /etc/mysql/mysql.conf.d/mysqld.cnf
- Use lineinfile Ansible module to change the relevant line
- In order to find the relevant line use regexp: 'bind-address\s*='
- Replace the bind address with *
- Create a handler to restart the MySQL service

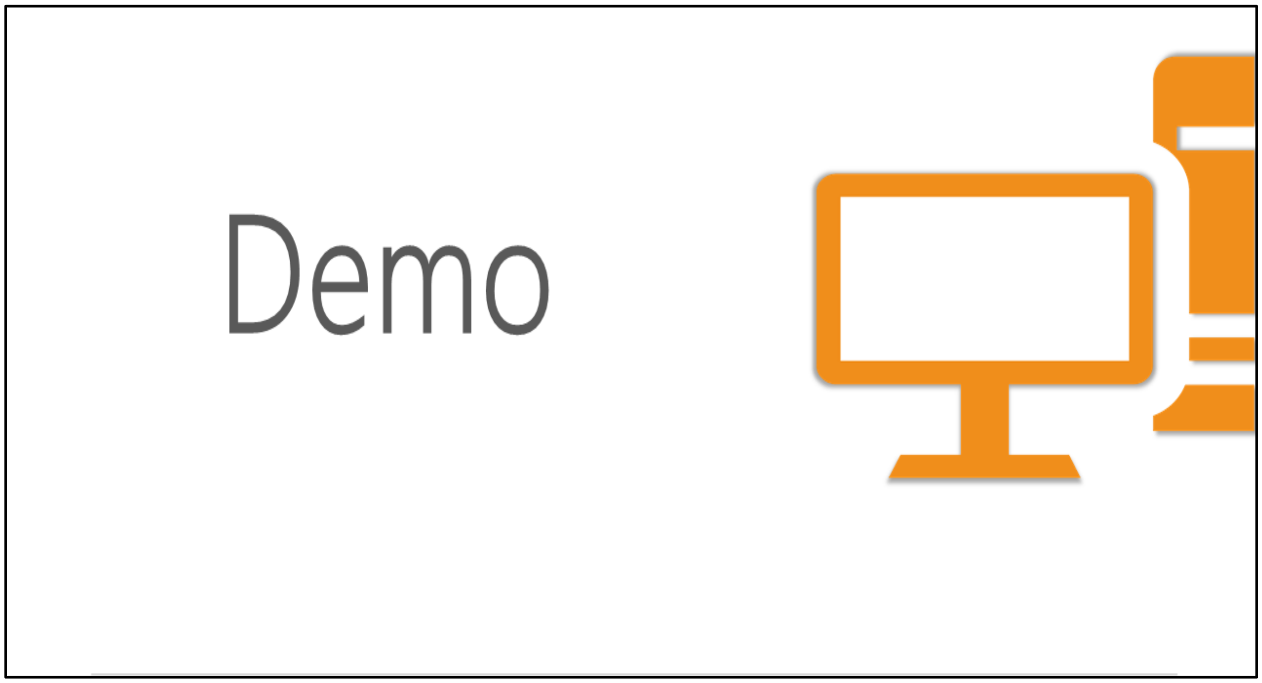
## Lab: Lab 6: Install MySQL Client On Webservers

## Install MySQL Client On Webservers



- Use apt module
- Install mysql-client on webservers group

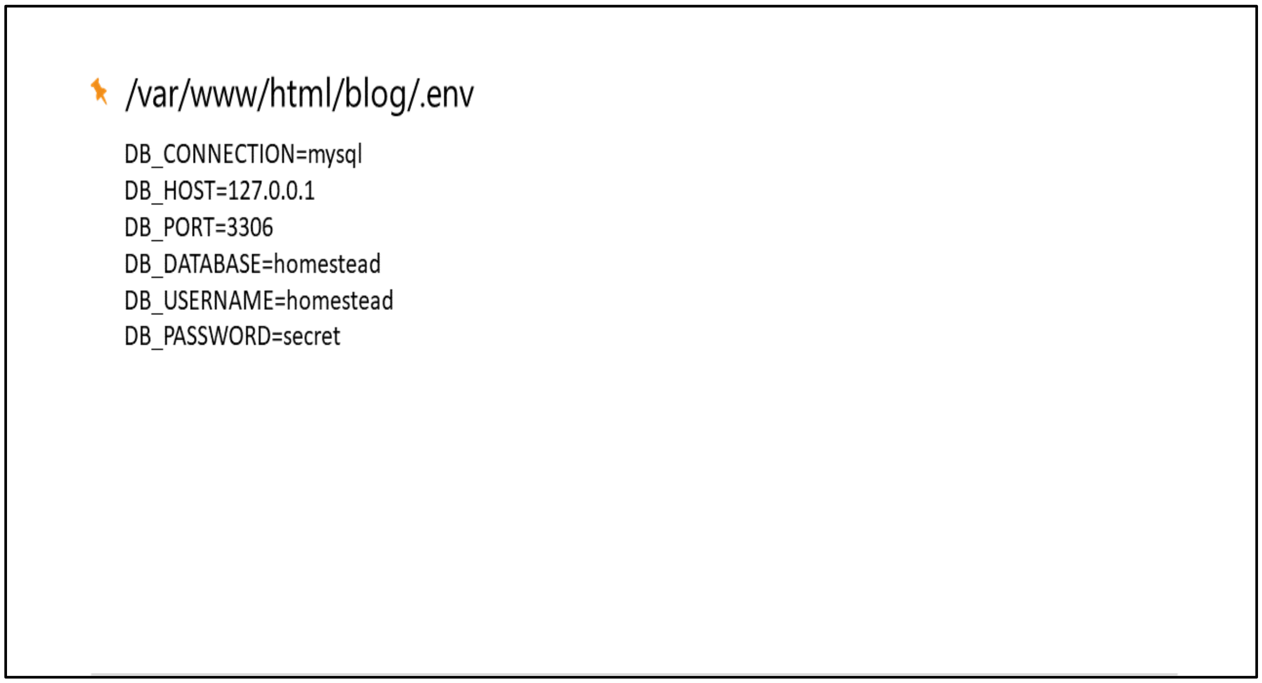
## Demo7: Create Laravel DB



Demo

## Demo8: Create Users Table

## Configure Laravel

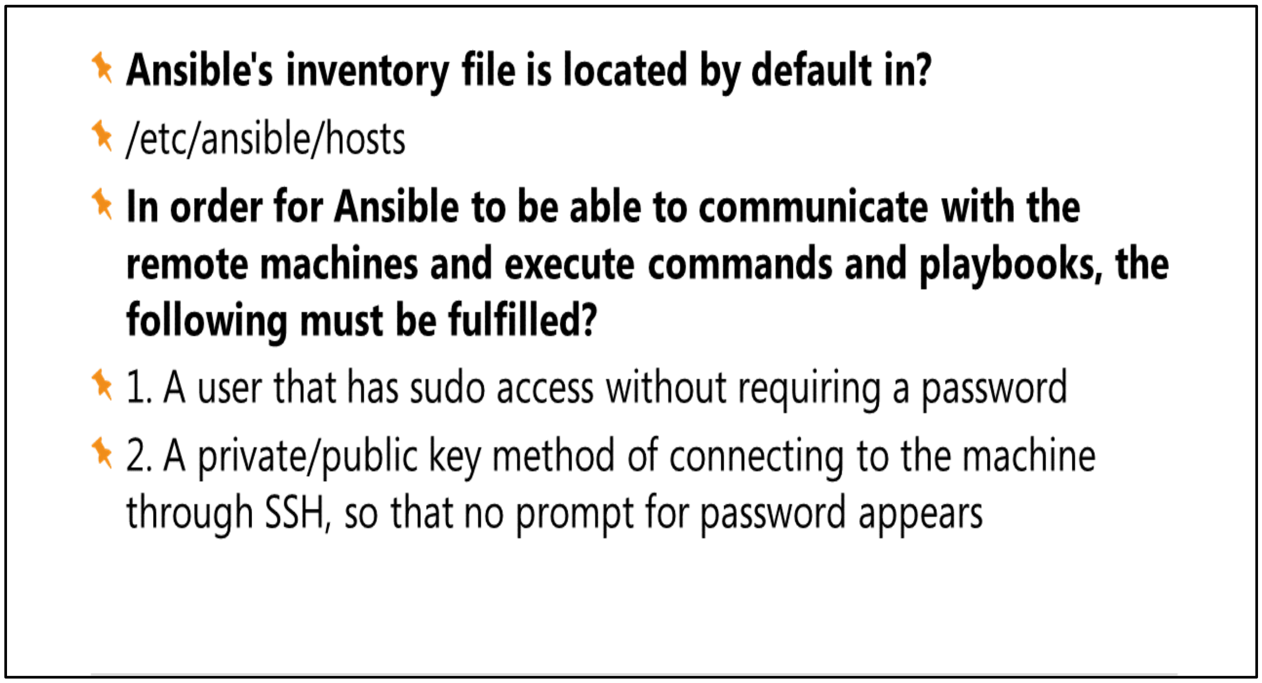


- /var/www/html/blog/.env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

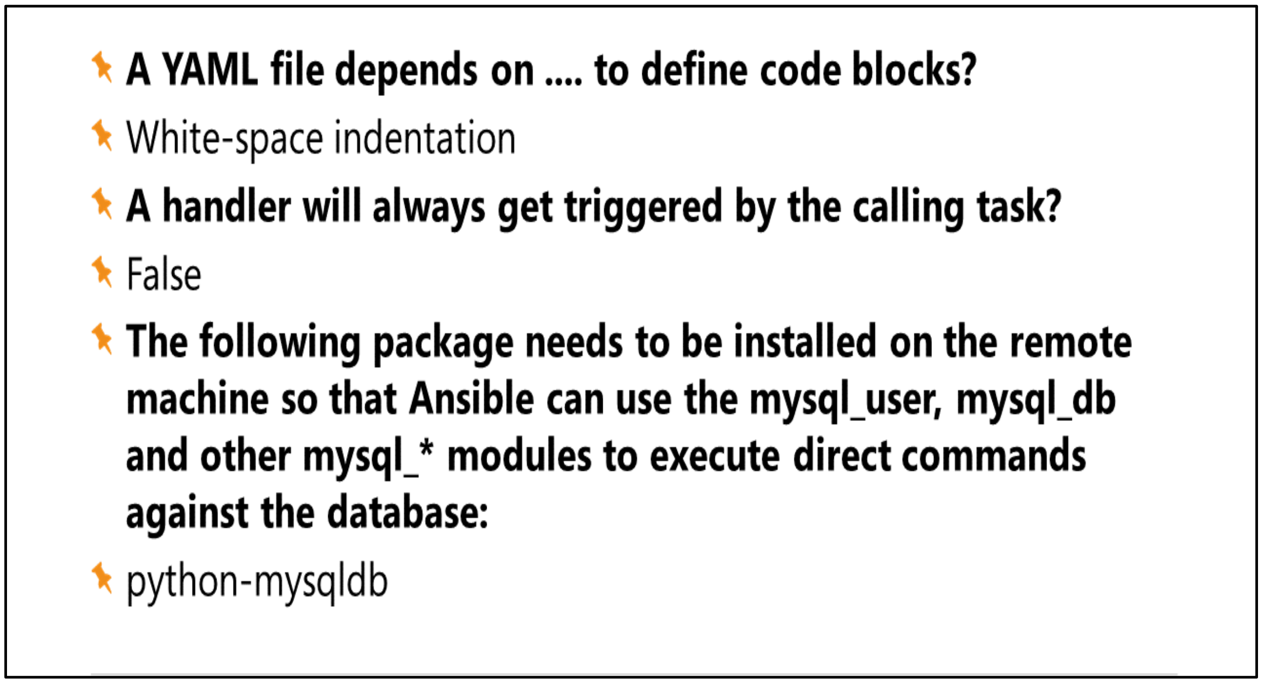
## Demo9: Configure Laravel

## Quiz



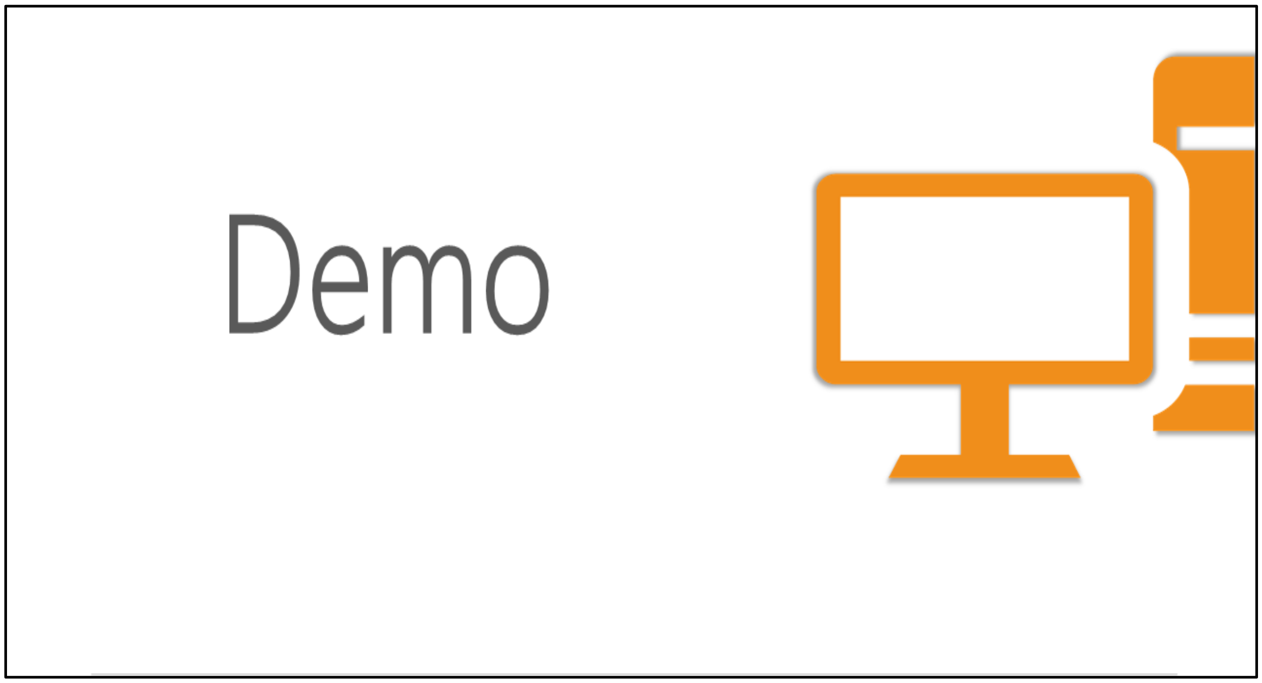
- **Ansible's inventory file is located by default in?**
- /etc/ansible/hosts
- **In order for Ansible to be able to communicate with the remote machines and execute commands and playbooks, the following must be fulfilled?**
- 1. A user that has sudo access without requiring a password
- 2. A private/public key method of connecting to the machine through SSH, so that no prompt for password appears

## Quiz □



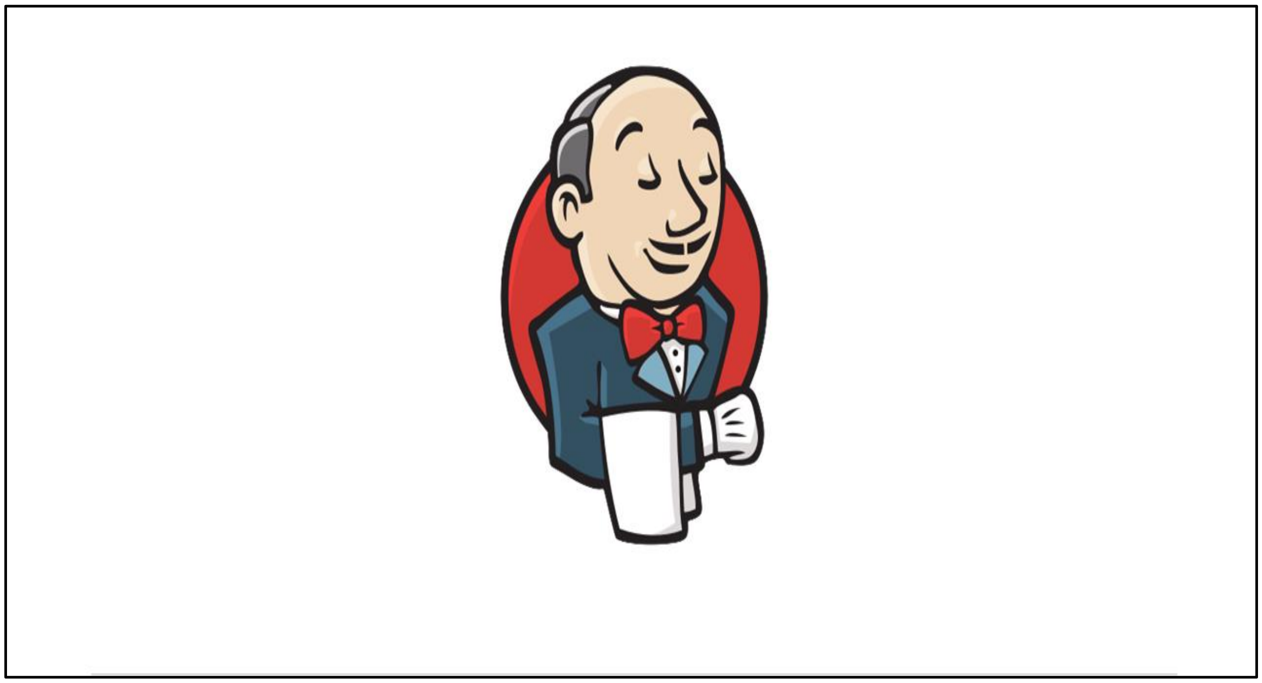
- **A YAML file depends on .... to define code blocks?**
- White-space indentation
- **A handler will always get triggered by the calling task?**
- False
- **The following package needs to be installed on the remote machine so that Ansible can use the mysql_user, mysql_db and other mysql_* modules to execute direct commands against the database:**
- python-mysqldb

## Demo10: Nginx Configuration

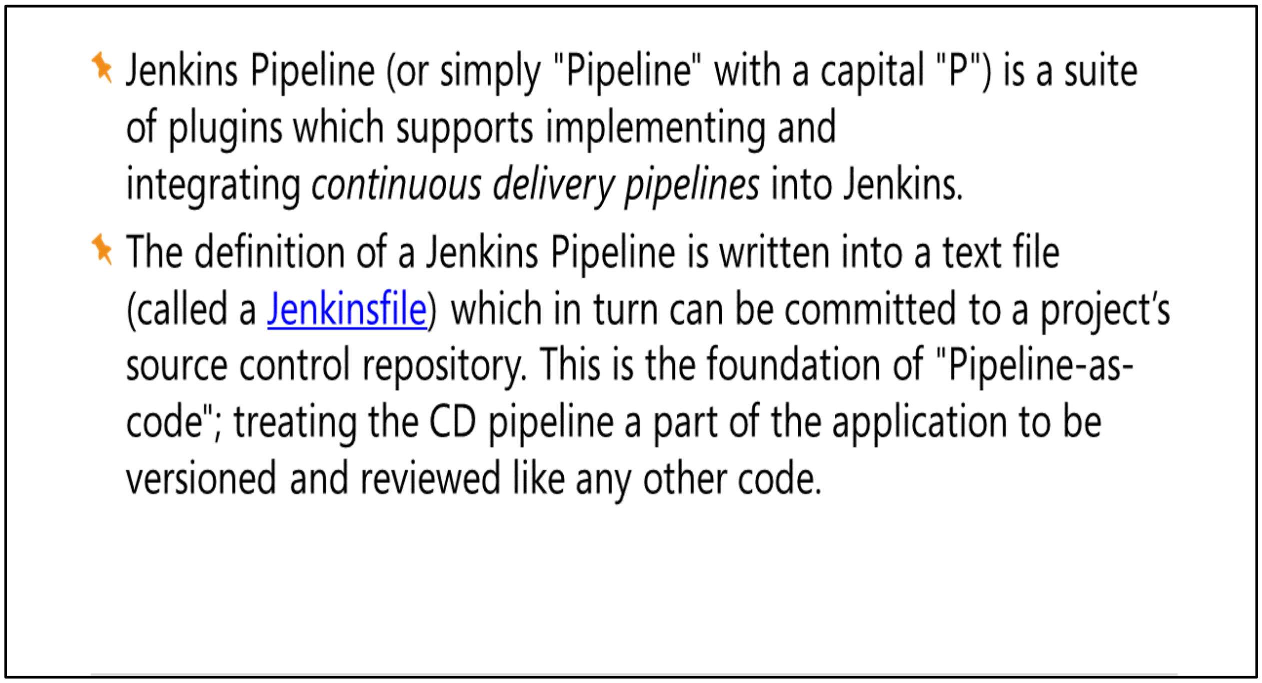


Demo

## Jenkins

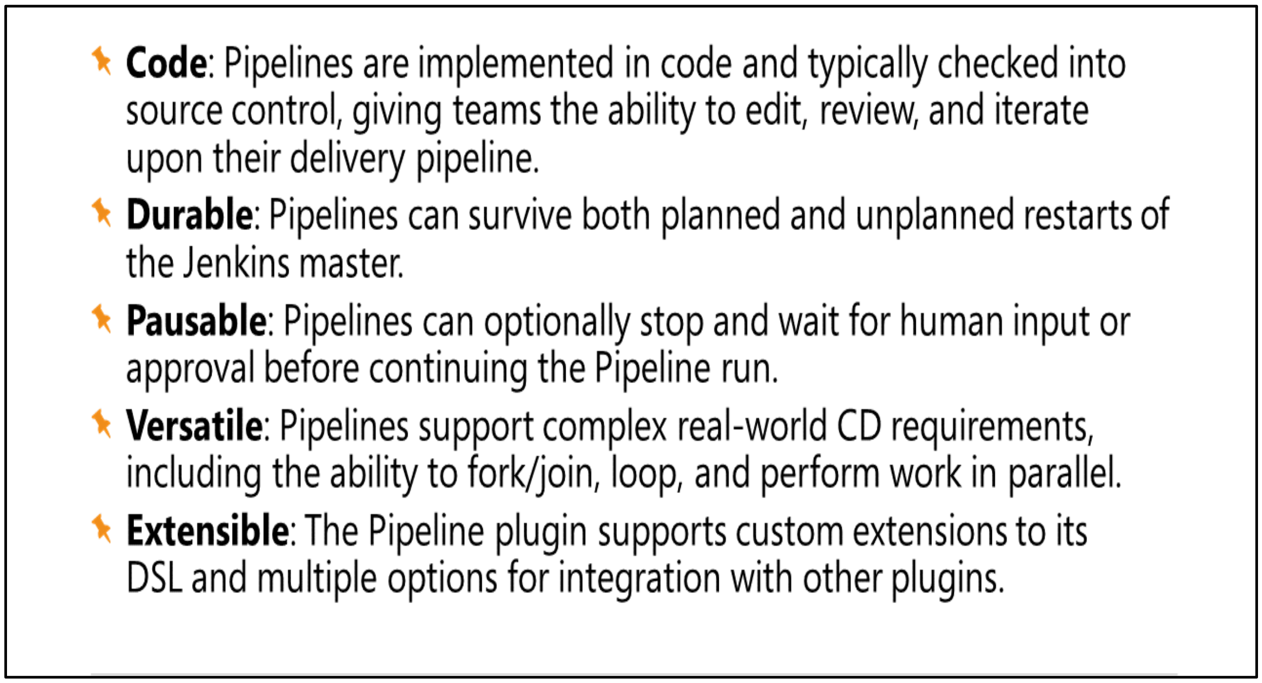


## What is Jenkins Pipeline?



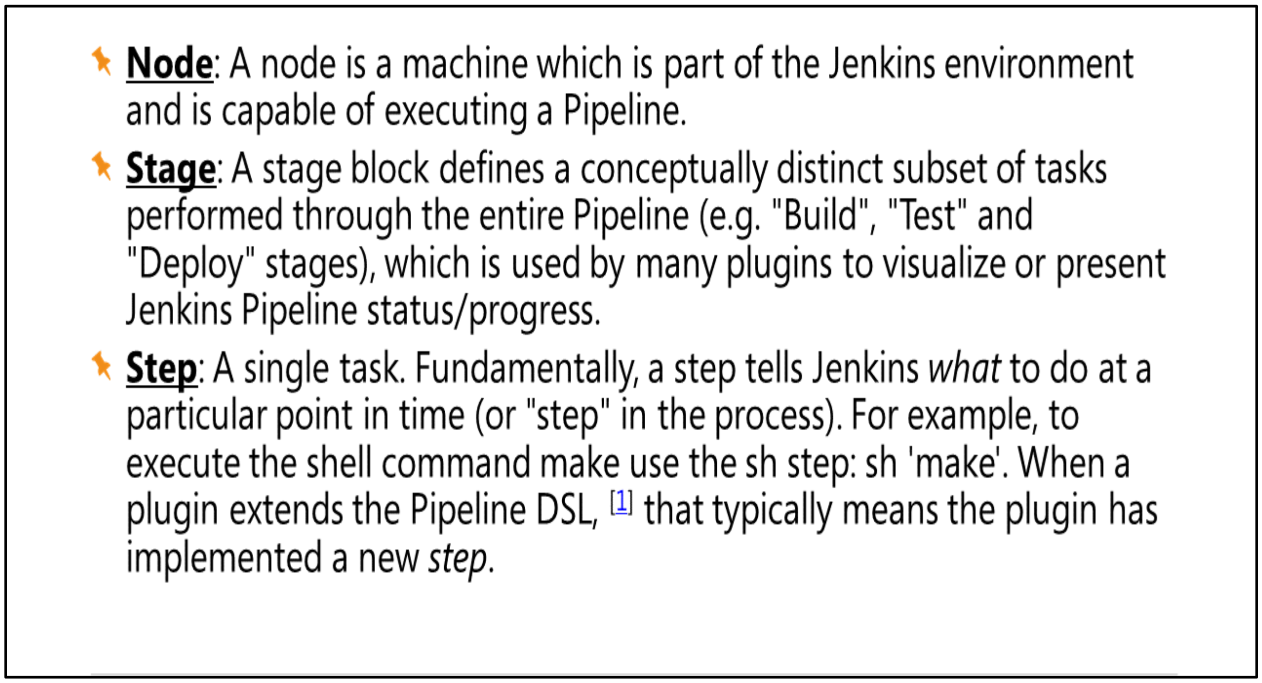
- Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.
- The definition of a Jenkins Pipeline is written into a text file (called a Jenkinsfile) which in turn can be committed to a project's source control repository. This is the foundation of "Pipeline-as-code"; treating the CD pipeline a part of the application to be versioned and reviewed like any other code.

## Why Pipeline?



- **Code**: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- **Durable**: Pipelines can survive both planned and unplanned restarts of the Jenkins master.
- **Pausable**: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- **Versatile**: Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- **Extensible**: The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.

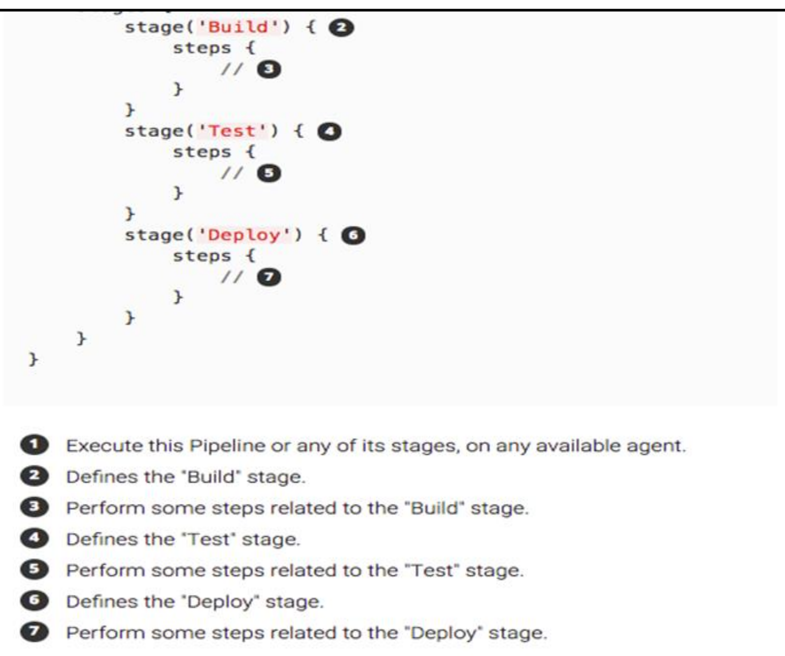
## Pipeline Concepts



- **Node**: A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.
- **Stage**: A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress.
- **Step**: A single task. Fundamentally, a step tells Jenkins *what* to do at a particular point in time (or "step" in the process). For example, to execute the shell command make use the sh step: sh 'make'. When a plugin extends the Pipeline DSL, [1] that typically means the plugin has implemented a new *step*.

## Declarative Pipeline fundamentals

- In Declarative Pipeline syntax, the pipeline block defines all the work done throughout your entire Pipeline.



```
            stage('Build') { ❷
                steps {
                    // ❸
                }
            }
            stage('Test') { ❹
                steps {
                    // ❺
                }
            }
            stage('Deploy') { ❻
                steps {
                    // ❼
                }
            }
        }
    }
```

❶ Execute this Pipeline or any of its stages, on any available agent.
❷ Defines the "Build" stage.
❸ Perform some steps related to the "Build" stage.
❹ Defines the "Test" stage.
❺ Perform some steps related to the "Test" stage.
❻ Defines the "Deploy" stage.
❼ Perform some steps related to the "Deploy" stage.

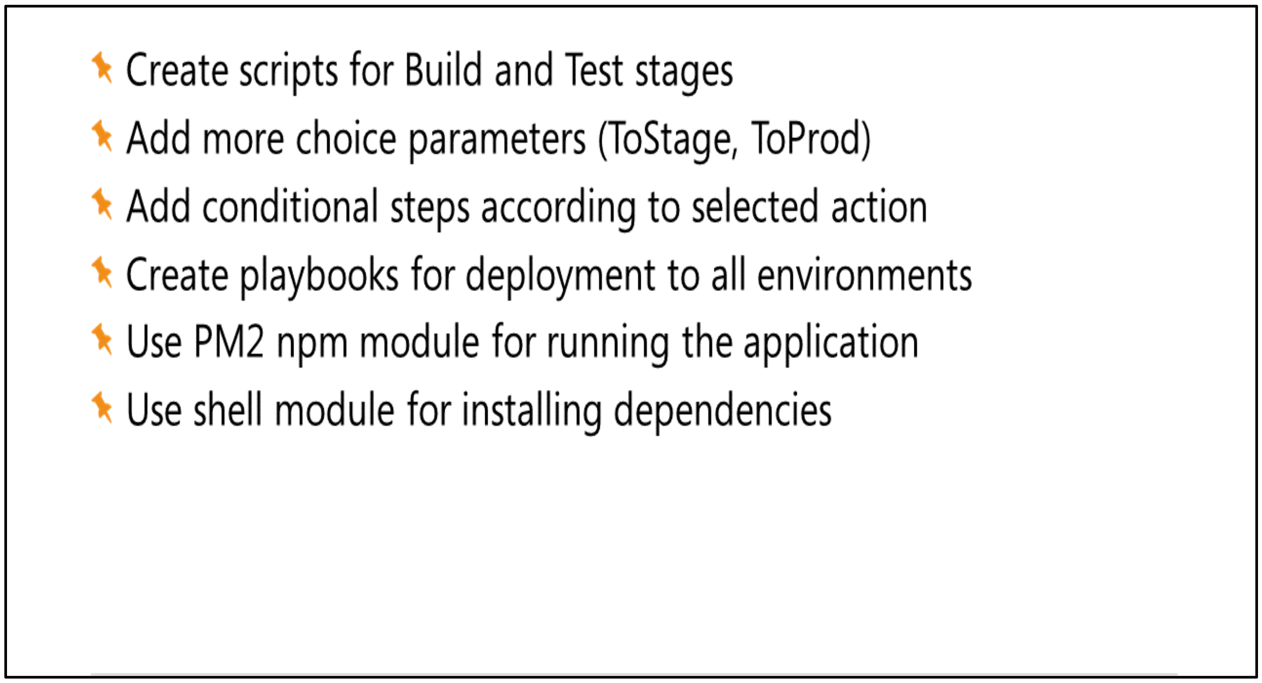
## Demo11: Create Blue Ocean Pipeline

Demo

Questions

## Lab: Lab 8: Create Full Pipeline

## Lab 8: Create Full Pipeline



- Create scripts for Build and Test stages
- Add more choice parameters (ToStage, ToProd)
- Add conditional steps according to selected action
- Create playbooks for deployment to all environments
- Use PM2 npm module for running the application
- Use shell module for installing dependencies