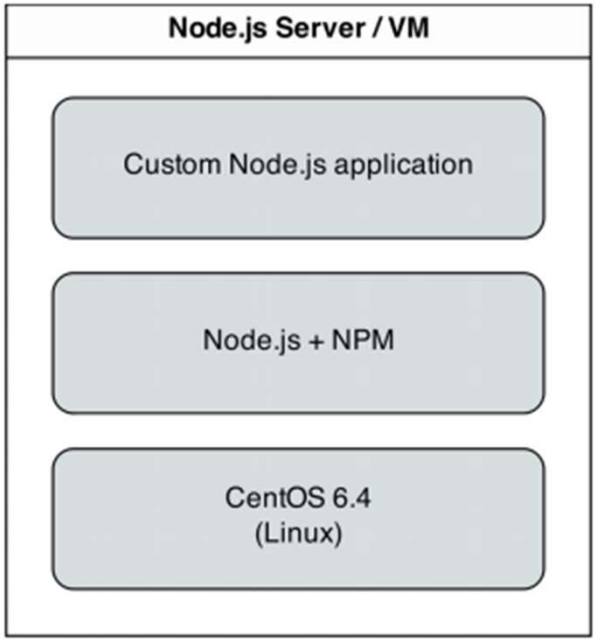
Ansible – Node js Playbook

Agenda

- Installing a Node JS
- Deploying a simple App
- Using variables
- Registering variables
- Skip tasks and –start-at-task option

Node JS App Server

- Lets run a more complex playbook
- Playbook will configure a CentOS server with Node.js, and install and start a simple Node.js application



Node.js app on CentOS

Add Repositories

- Add both the EPEL and Remi repositories, so we can get some packages like Node.js or later versions of other necessary software
- Shell script uses the rpm command to import the EPEL and Remi repository GPG keys, then adds the repositories, and finally installs Node.js

Add Repositories

```
# Import Remi GPG key - see: http://rpms.famillecollet.com/RPM-GPG-KEY-remi
wget http://rpms.famillecollet.com/RPM-GPG-KEY-remi \
  -O /etc/pki/rpm-gpg/RPM-GPG-KEY-remi
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi

# Install Remi repo.
rpm -Uvh --quiet \
  http://rpms.famillecollet.com/enterprise/remi-release-6.rpm

# Install EPEL repo.
yum install epel-release

# Install Node.js (npm plus all its dependencies).
yum --enablerepo=epel install node
```

Node JS contd..

- Ansible makes it more robust, though little verbose

```
---
- hosts: all

  vars:
    # Vars can also be passed in via CLI with `--extra-vars="name=value"`.
    node_apps_location: /usr/local/opt/node

  tasks:
    - name: Install Remi repo.
      yum:
        name: "http://rpms.remirepo.net/enterprise/remi-release-7.rpm"
        state: present

    - name: Import Remi GPG key.
      rpm_key:
        key: "http://rpms.remirepo.net/RPM-GPG-KEY-remi"
        state: present

    - name: Install EPEL repo.
      yum: name=epel-release state=present

    - name: Ensure firewalld is stopped (since this is a test server).
      service: name=firewalld state=stopped

    - name: Install Node.js and npm.
      yum: name=npm state=present enablerepo=epel
```

Node JS contd..

```
- name: Install Forever (to run our Node.js app).
  npm: name=forever global=yes state=present

- name: Ensure Node.js app folder exists.
  file: "path={{ node_apps_location }} state=directory"

- name: Copy example Node.js app to server.
  copy: "src=app dest={{ node_apps_location }}"

- name: Install app dependencies defined in package.json.
  npm: "path={{ node_apps_location }}/app"

- name: Check list of running Node.js apps.
  command: forever list
  register: forever_list
  changed_when: false

- name: Start example Node.js app.
  command: "forever start {{ node_apps_location }}/app/app.js"
  when: "forever_list.stdout.find(node_apps_location + '/app/app.js') == -1"
```

PlayBook Walkthrough

- Install the Remi Repo using the Yum module
- rpm_key is a very simple Ansible module that takes and imports an RPM key from a URL or file, or the key id of a key that is already present, and ensures the key is either present or absent (the state parameter)
- yum installs the EPEL repository
- Since this server is being used only for test purposes, we disable the system firewall so it won't interfere with testing
- yum installs Node.js (along with all the required packages for npm, Node's package manager) if it's not present, and allows the EPEL repo to be searched via the enablerepo parameter (you could also explicitly disable a repository using disablerepo)
- Since NPM is now installed, we use Ansible's npm module to install a Node.js utility, forever to launch our app and keep it running. Setting global to yes tells NPM to install the forever node module in /usr/lib/node_modules/ so it will be available to all users and Node.js apps on the system.

Deploy Node JS app

- Next step is to install a simple Node.js app on our server.
- First, we'll create a really simple Node.js app by creating a new folder, app, in the same folder as your playbook.yml.
- Create a new file, app.js, in this folder, with the following contents

Deploy Node JS app

```
// Simple Express web server.
// @see http://howtonode.org/getting-started-with-express

// Load the express module.
var express = require('express'),
app = express.createServer();

// Respond to requests for / with 'Hello World'.
app.get('/', function(req, res){
    res.send('Hello World!');
});

// Listen on port 80 (like a true web server).
app.listen(80);
console.log('Express server started successfully.');
```

Deploy Node JS app contd..

- Since this little app is dependent on Express (a simple http framework for Node), we also need to tell NPM about this dependency via a package.json file in the same folder as app.js

```
{
  "name": "examplenodeapp",
  "description": "Example Express Node.js app.",
  "author": "Your Name and Email",
  "dependencies": {
    "express": "3.x.x"
  },
  "engine": "node >= 0.10.6"
}
```

PlayBook – Walkthrough contd..

- First, we ensure the directory where our app will be installed exists, using the file module. The {{ node_apps_location }} variable used in each command can be defined under a vars section at the top of our playbook, in your inventory, or on the command line when calling ansible-playbook
- Second, we copy the entire app folder up to the server, using Ansible's copy command, which intelligently distinguishes between a single file or a directory of files, and recurses through the directory, similar to recursive scp or rsync
- Third, we use npm again, this time, with no extra arguments besides the path to the app. This tells NPM to parse the package.json file and ensure all the dependencies are present

PlayBook – Walkthrough contd..

- Launching the app
- register creates a new variable, forever_list, to be used in the next play to determine when to run the play. register stashes the output (stdout, stderr) of the defined command in the variable name passed to it
- changed_when tells Ansible explicitly when this play results in a change to the server. In this case, we know the forever list command will never change the server, so we just say false— the server will never be changed when the command is run
- Once the playbook has finished configuring the server and deploying your app, visit http://hostname/ in a browser (or use curl or wget to request the site)

PlayBook – Walkthrough contd..

- The second play actually starts the app, using Forever.
- We could also start the app by calling node {{ node_apps_location }}/app/app.js, but we would not be able to control the process easily, and we would also need to use nohup and & to avoid Ansible hanging on this play
- Forever tracks the Node apps it manages, and we use Forever's list option to print a list of running apps. The first time we run this playbook, the list will obviously be empty—but on future runs, if the app is running, we don't want to start another instance of it. To avoid that situation, we tell Ansible when we want to start the app with when. Specifically, we tell Ansible to start the app only when the app's path in not in the forever list output

Skip tasks -

- Sometimes in long playbooks, you may want to skip some tasks
- You can use the –start-at-task option to skip tasks

- ansible-playbook nodejs.yml --start-at-task="Install Node.js and npm.“

- Assignment : Write playbook for stopping the nodejs application