PROJECT

AUTOMATION FOR SOFTWARE DEVS

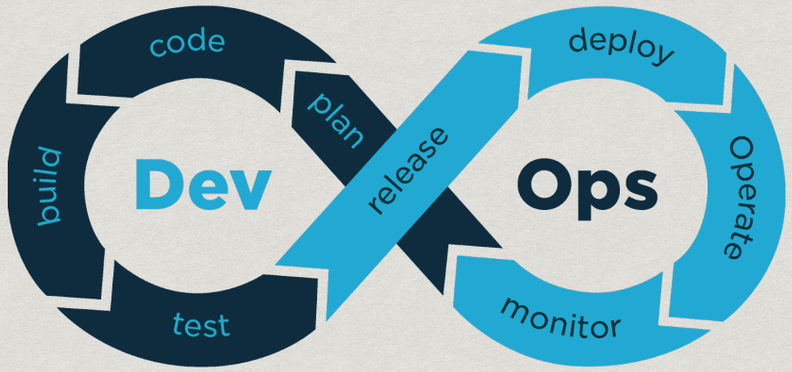
By Edison Hanchell

DATE 18/01/2020

CLIENT NASSAU TECH MEETUP

Who am I?

- Software Developer
- DevOps Engineer
- Virtualization Expert
- Educator



What I do?

- I write code and then write the code that tests the code so that the other code I write can get the code to where it needs to be.
- I also cook.

Languages

- Java
- Javascript
- C#
- Python
- Powershell
- Bash
- Ruby
- PHP
- DOS
- Actionscript
  (Please don’t use this anymore)

AUTOMATION

What is automation?

- The Google tells us that:
  Automation is the technology by which a process or procedure is performed with minimal human assistance

I don't necessarily like that definition because while true, it diminishes entirely what is actually happening. Automation is a way for a human to structure a process in their mind and frame it so that a computer or some other machine is able to understand it. It's us speaking directly to the machine and it letting us know it understands. I say understands because if it doesn't it will sure as shit let you know.

Throughout the talk, I'm going to go over a few technologies that vary in complexity to accomplish the same or similar results. I will preface this by saying that this is **Not at all** the only way to accomplish these tasks. There are methods galore but it is a nice introduction or a fun distraction (depending on your level) to add some automation to your day and maybe even workflow.

CODING

Demo: 01-Gulp

I'm going to go off of the highly probable assumption that about 80% of the room uses Visual Studio Code. I was a Sublime Text advocate for the longest but eventually made the switch multiple times but after the last one I never looked back. VS is just right....that is all. No but In almost every language but unfortunately not all, I use it religiously because the features that I need are there and the ones that I want can be downloaded or made.

Gulp
When writing, we all use a linter to check our code in realtime so I'm not gonna bother with discussing syntax but what I will go over is formatting in post. One of the tools that I use today whenever webpack isn't available is Gulp.

CRONTAB

Demo: 02-Crontab

The built-in Linux Crontab is a scheduler. Plain and simple. You can use it to set up regular whatever-the-hell you want. It's most often used to either monitor something at a regular interval and then tie into some script if a condition is met or carry out a regular task like cycling logs.

GIT

Demo: 03-Git

Git is a tool everyone is familiar with but there are some lesser known functionalities that make it an extraordinary automation tool if used correctly. That power lies in git hooks. The hooks that come included with any repository can be leveraged as a part of your project to perform tasks before or after specific actions are carried out.

Let's look at two:

* commit-msg - runs during commit and can be used to validate commit message
* post-receive - runs after commit is successful and can kick off a job

ANSIBLE

Demo: 04-Ansible

Ansible Demo Commands
Inventory File
The first thing to be done is to create an inventory file and place the required information about your hosts into it. We will define only a single host in this one.
```text
192.168.1.2
192.168.1.3
```

Then we add some variables to our host that can be used by any of the playbooks we trigger. Variables in this fashion are added onto the same line and space-delimited. We'll name the file `inventory`.
```text
192.168.1.2 sys_name=server1 fq_sys_name="{{ sys_name }}.domain.com"
192.168.1.3 sys_name=server2 fq_sys_name="{{ sys_name }}.domain.com"
```

Ansible Configuration
The next thing to do is to configure your Anisble run defaults file. This file is named `ansible.cfg` and is stored in the root location of the project. Notice how the inventory to be used is defined here.
```conf
[defaults]
```

inventory = ./inventory
remote_user = developer
deprecation_warnings = false

[privilege_escalation]
become = false
become_exe = sudo -i
become_user = root
```
View the included `ansible.cfg` file for more information on what each item does.

### Playbooks
The power of Ansible is unleashed through playbooks. These files define a series of tasks that can be performed against items within your inventory using the different modules that Ansible is comprised of. Each task that is used as well as available ones can be found within the Ansible documentation. A more in-depth explanation and other examples can be found here: [Intro to Playbooks]( https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html )
```

Running a Playbook

- When running an Ansible playbook, you receive a readout of the status of each task as it is ran. In the areas where you see '**changed**', that Ansible task was ran.
- In the areas that you see '**ok**' that task was not needed as the setting already exists.

```
edisonhanchell @ connovar in /Users/edisonhanchell/De
› ansible-playbook playbooks/pb_configure_ssh.yml -u root

PLAY [Configure SSH] ***********************************

TASK [Back SSHD Config] ********************************
changed: [67.207.90.146]

TASK [Disable Password Authentication] *****************
ok: [67.207.90.146]

TASK [Disable Root Login] ******************************
changed: [67.207.90.146]

RUNNING HANDLER [restart_ssh] **************************
changed: [67.207.90.146]

PLAY RECAP *********************************************
67.207.90.146              : ok=4    changed=3    unreachable=
```

Becoming and Conditionals

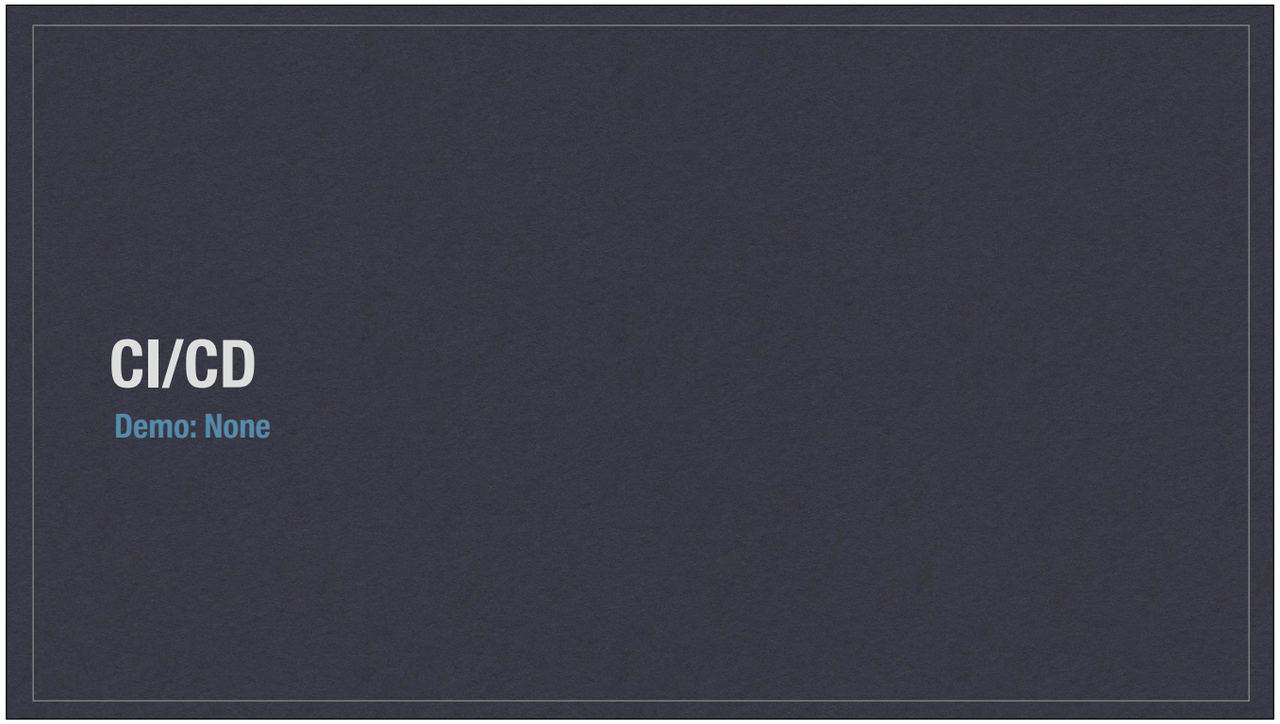
- When ran without the root user escalating permissions is a requirement. This is accomplished by using the '**-K**' flag. The result can be seen as it prompts the user for their password.
- In the area that says **skipping**, that task was not ran as there was a criteria in the playbook task that this host did not meet.

```
edisonhanchell @ connovar in /Users/edisonhanchell/Desktop/Automation for
❯ ansible-playbook playbooks/pb_set_hostname.yml -K
BECOME password:

PLAY [Initialize Server] ***********************************************************

TASK [Gathering Facts] *************************************************************
ok: [67.207.90.146]

TASK [Disable hostname reset on reboot] ********************************************
changed: [67.207.90.146]

TASK [Configure Hostname] **********************************************************
changed: [67.207.90.146]

TASK [Configure consistent hostname on Ubuntu for demo] ****************************
skipping: [67.207.90.146]

TASK [Configure consistent hostname on Ubuntu for demo.loborenard.com] *************
changed: [67.207.90.146]

TASK [Reboot to confirm changes] ***************************************************
changed: [67.207.90.146]

PLAY RECAP *************************************************************************
67.207.90.146              : ok=5    changed=4    unreachable=0    failed=0    ski
```

PARALLELS

Demo: 05-Parallels

Parallels is a tool made by GNU that allows you, with minimal effort to run commands concurrently or as the name suggests, in parallel. The way it works is that you pass in the number of iterations as well as any parameters for each run on a single line and then pass that to parallels and give it the command that you want it to run against this data. I have personally used this for heavy testing on any site or system that I'm using.



Next, is the heaviest tool that I'll be demonstrating. Gitlab. Gitlab is essentially Github but not public. What I'll be showing you is one of the critical features for any functional software development team and that is its Continuous Integration and Continuous Deployment Workflow. More information on the CI/CD process in Gitlab can be found at https://docs.gitlab.com/ee/ci/README.html.