Local Development Databases

Wes McClure
wesmcclure.com

pluralsight
hardcore dev and IT training

Why Manage Databases With Vagrant?

- Databases are resource hogs
- Difficult setup
- Difficult configuration
- Project specific configuration
- OS differences
- Load datasets on demand
- Enables local development databases

Additional Learning

- Provisioners
- Puppet
- Desired state configuration
- vagrant package
- vagrant global-status

Provisioning

- **Shell provisioning**
  - Easy starting place
  - Grows unruly
- **Sys admin pain**
- **Configuration Management**

Provisioners

- **Out of box**
  - Shell
  - File
  - CM Tools
    - Puppet agent and apply
    - Chef client and solo
    - Salt
    - Ansible
    - CFEngine
    - Docker
- **Plugins**
- **Mix and match**

Provisioners Applied During:

- First vagrant up
- vagrant provision
- vagrant reload --provision
- vagrant up --provision
- vagrant up --no-provision
- run: “always”

Why Puppet?

- Maintain desired state
- Software, OS features, configuration
- Community of modules
- Idempotent – desired state
- Vagrant - learn CM

Why Puppet?

- Maintain desired state
- Software, OS features, configuration
- Community of modules
- Idempotent – desired state
- Vagrant - learn CM

Overview

- Add mongodb box
- Configure three machines
- Launch replica set
- Replicate zip data set
- Secondary data loss
- Primary failure

Reflect

- vagrant destroy – backup
- vagrant up – restore
- Restore from other developers
- Restore from backups
- Test migrations
- Continuous integration
- Save resources