SAND2017-5348C

https://i1.wp.com/www.ryadel.com/wp-content/uploads/2015/08/vmware_vsphere.png?resize=750%2C410

An approach to Air-gapped Deployment

Joseph Gonzalez

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

What Am I Covering?

- How we are deploying Chef in an air-gapped environment
  - *(Guidance from a couple of smart Chefs)*
- The development process(currently)
- How/Why

How Am I Covering

- Who Am I?
  - About me
- Our Project
- Architecting The Solution
  - ***Organization Deployment(Chef-Countertop)***
  - ***Cookbook Pipeline***
- Related Development
- Advice For New Developers
- Closing

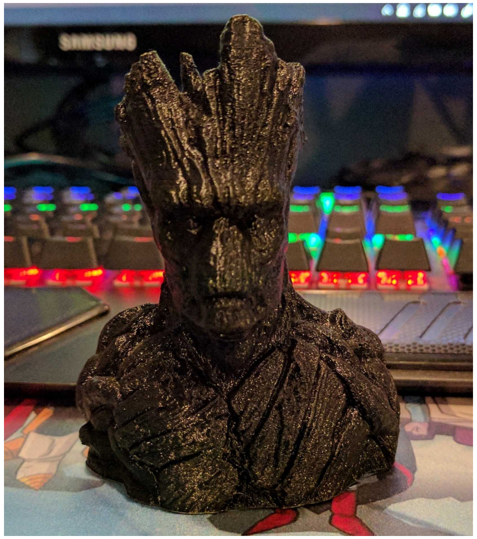
WHO AM I?

About me

- From California
- Bachelors in Computer Engineering
- California State University Sacramento
  - Alumni 2016
- Professional Experience
  - Hewlett Packard Enterprise
  - Sandia National Laboratories (Solutions Architect)
- New to Devops
- New to Chef

Interests

- 3D printing
- Arduino, Pi, CNC
- Building software
  - Apps, games, scripting
- Lifelong student



What is Sandia?

- Part of the U.S. Laboratory System (NNSA)
  - NM, CA
  - WWII Manhattan project (1948)
- Government agencies, industry, and academic institutions
- Primary Mission: "***The synergy and interdependence between our nuclear deterrence mission and broader national security missions forge a robust capability base and empower us to solve complex national security problems***" – Sandia
- Missions that support national security

LETS BEGIN

Why Does This Matter

- Deployment
  - Troublesome – Internet Dependent
  - ChefDK, Chef-Server, Chef-Node packages
- Target environments are restricted
- Corporate restrictions
  - Proxy
  - Certificates
  - Limited Internet
  - Air-gapped networks

Why Does This Matter

- Automation & Configuration management
  - ease system setup
  - Adds visibility
  - Adds traceability (as code)
  - Can add network dependencies
    - No internet access!!
- Issues moving Infrastructure code
  - Development - > Production
- No magic, things don’t just work out
  - How do we get to were we want to be?
    - Automate configuration, install packages, setup our systems(machines)

OUR PROJECT

Project Deliverable

- Full stack delivery
  - infrastructure-> runtime->applications
- Machines
  - Run in a private facility
  - Air-gapped environment
  - Build traceability
  - No manual machine installation
  - Code traceability
- Installed on-site
- No internet delivery

Why did we choose Chef?

- Complex
  - Installation and software need automation
- Operations
  - System state
    - Keeps systems in predictable states
    - Test environments confidence
    - Reproducible machine states
- Two forms of delivery
  - Machines as Chef Orgs (Large updates/Machine VM upgrades, NICS etc.)
  - Cookbooks to upload (minor updates)
- REPODUCIBILITY REPODUCIBILITY REPODUCIBILITY REPODUCIBILITY !!

My Role

- To Architect the way we use Chef
- Automate with Chef & Pipelines (CI/CD)
- Probably more roles ☺

Obvious Issues

- Chef Components
  - Provisioning Gems
    - Disparate workstations
- Putting an Org together
  - Bootstrapping w/knife requires internet connection
    - Proxy is now an issue
  - SSL issues

Getting around issues

- Assume air-gapped
  - Stop “insecure”
  - Stop adding your internal proxies to deliverable products! (Don’t do that)
- For recipes
  - Dependencies into cookbook
    - Simplify (No Berkshelf)
    - Don’t hand off responsibility
    - Avoid resources using drivers

ARCHITECTING THE SOLUTION

Components of Development

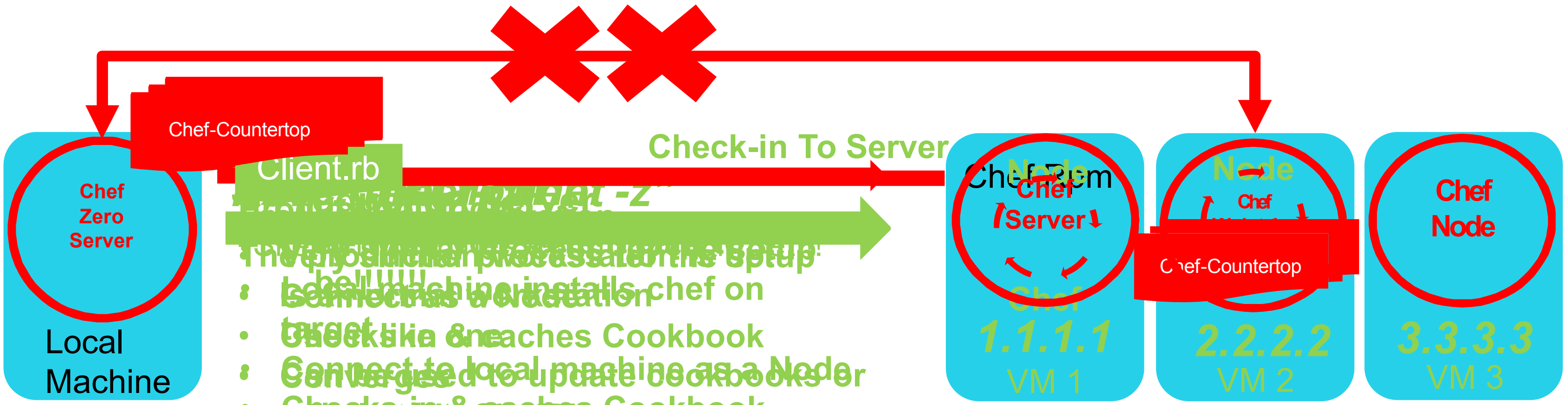
1. Chef-Countertop (Deployer)
2. Cookbook Pipeline

Chef-Countertop (Deployer)

- Goal
  - Turns a machine into a chef machine(server, node, workstation)
- Includes packages
  - server, node, chefDK artifacts
  - Artifacts inside cookbook
  - Doesn’t rely on Artifactory/Nexus/FTP Servers
    - For now…
- Comparable solution to what is shown by Chef
- Utilizes chef’s “machine” resource
  - Calling options for “converge_only”
    - ^ Does not install chef

CHEF-COUNTERTOP

Chef-Countertop



[illegible]

- [illegible] installs chef on target
- Checks in & caches Cookbook
- Connect to local machine as a Node
- Checks in & caches Cookbook
- 1st Convergence
  - Configures [illegible]
  - Client.rb → New Server
  - Cuts the cord(Client)
  - Connects to new server
  - Grabs Productions artifacts from cache
- 2nd Convergence
  - Pushes Production Cookbooks
  - Pushes Production Policies
  - Executes its production role

Converging…

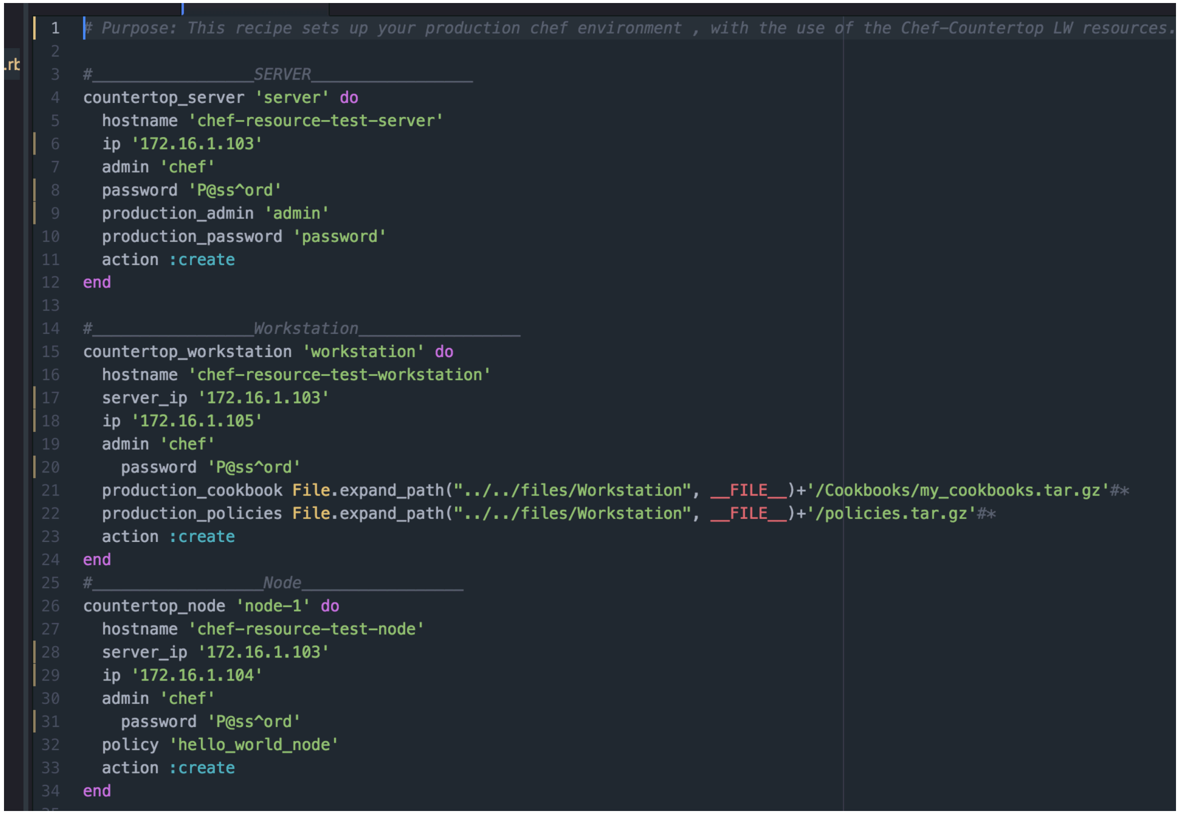
- Installs Chef Server
- Configures Chef Server
- Cut the Cord (Cli

Production Cookbooks

Policy FIles

Chef-Countertop

- Setup to be a custom resource
- Gets called from an organization cookbook
- Defines your project setup



```ruby
# Purpose: This recipe sets up your production chef environment , with the use of the Chef-Countertop LW resources.

#_________________SERVER_________________
countertop_server 'server' do
  hostname 'chef-resource-test-server'
  ip '172.16.1.103'
  admin 'chef'
  password 'P@ss^ord'
  production_admin 'admin'
  production_password 'password'
  action :create
end

#_________________Workstation_________________
countertop_workstation 'workstation' do
  hostname 'chef-resource-test-workstation'
  server_ip '172.16.1.103'
  ip '172.16.1.105'
  admin 'chef'
    password 'P@ss^ord'
  production_cookbook File.expand_path("../../files/Workstation", __FILE__)+'/Cookbooks/my_cookbooks.tar.gz'#*
  production_policies File.expand_path("../../files/Workstation", __FILE__)+'/policies.tar.gz'#*
  action :create
end
#__________________Node_________________
countertop_node 'node-1' do
  hostname 'chef-resource-test-node'
  server_ip '172.16.1.103'
  ip '172.16.1.104'
  admin 'chef'
    password 'P@ss^ord'
  policy 'hello_world_node'
  action :create
end
```

Lessons Learned

- Chef-Ingredient Cookbook
  - Internet
  - SRC similar process
- Creating machines
  - Easier to have machines pre-provisioned (stood-up)
  - Vagrant-Vmware adds more layers
    - Chef calls Vagrant -> Vagrant -> Vmware
    - Hand-off
    - Dissimilar result machines are great for deploying to different environments
      - Dockers, AWS…
    - Not necessary

COOKBOOK PIPELINE

Cookbook Pipeline

- Gitlab-Runner
- 3 Classes of cookbooks

Classes

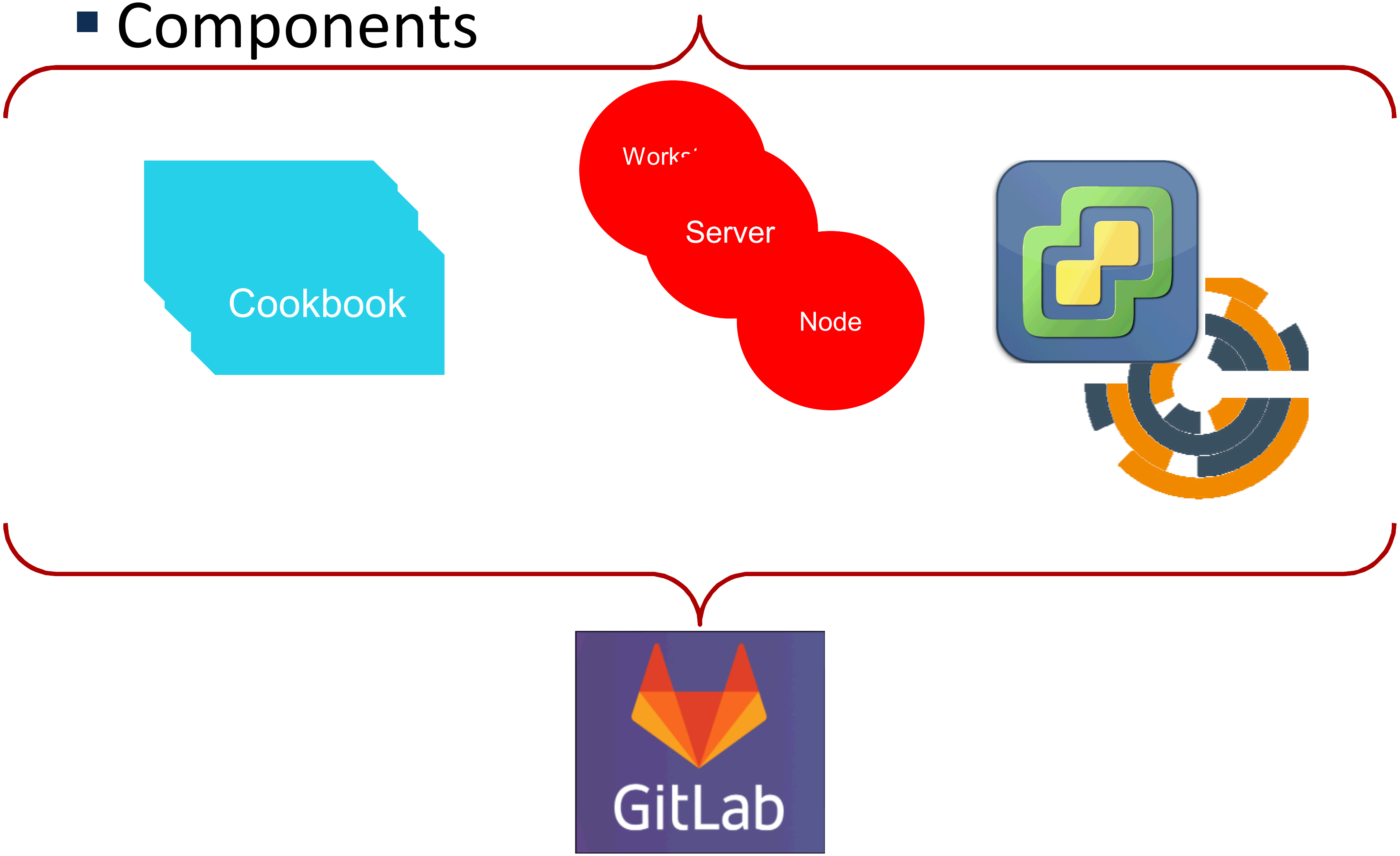
Resource Cookbook

Role Cookbook

Organization Cookbook

Cookbook Pipeline

- Components



Cookbook

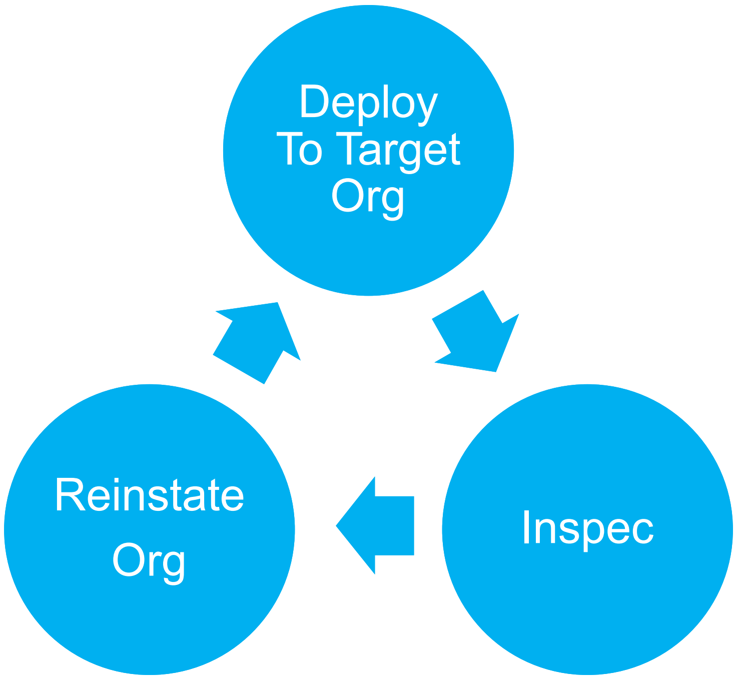
Works

Server

Node

GitLab

Cookbook Pipeline



- Code is pushed
- GitLab-Runner picks up cookbook
- Runner uses Knife-Vsphere to clone
- running chef-org
- Turns the original chef-org off
- Uploads cookbool server
- Places policyfil
- Invokes Run

- Knife-Vsphere destroy temp org if successful
- Knife-Vsphere Boot up original chef-org
- Was this a successful run ?

- Runner runs In node
- Do they pass

Upload to Chef-Server

Lessons Learned

- Machine Persistence
    - Running cookbooks on the pipeline org puts the machine into unknown state
    - Kitchen style provisioning
        - Create machine, run cookbook , destroy it
        - Fresh machine, in a known state
        - Larger scale w/Vmware Vsphere using special images similar to production images
- Cookbook Dependencies
    - Role cookbooks have dependencies
        - 1 chef server for all of our work
        - Validated -> push
        - Cloning a real org eliminates the dep problem (Custom cookbooks)
            - The cloned server now has all the cookbooks that have passed the pipeline
            - No supermarket (Needs more investigation)
    - Dependency Management !!

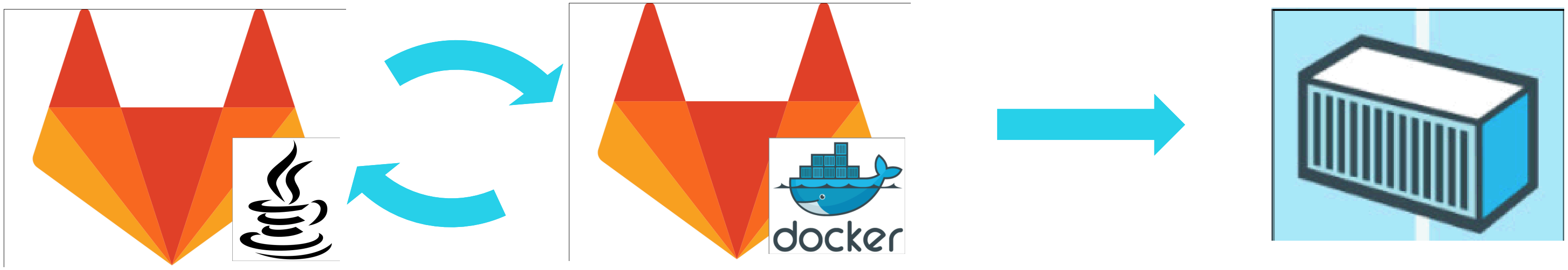
Lessons Learned

- Enforce Cookbook order
  - Cookbook dependent order
  - Cookbook needs to exist on server
- Separate developers from accessing chef server
  - People are not uploading broken code (layer of safety)
  - No need to link knife to a server
    - Easier because we have proxies and certificate authorities that bypass
- Knife-Vsphere
  - Very handy
  - Easy to script from gitlab-runner
  - Recommended for provisioning datacenters

RELATED DEVELOPMENT

Application Pipeline

- Gitlab
- 2 Repos per application component
- Complimentary repos work together
- Automated build process
  - Committing code to application repo triggers new Docker image builds
  - Passes build artifacts
  - Cross project pipelining



App Repo

Docker Repo

Container

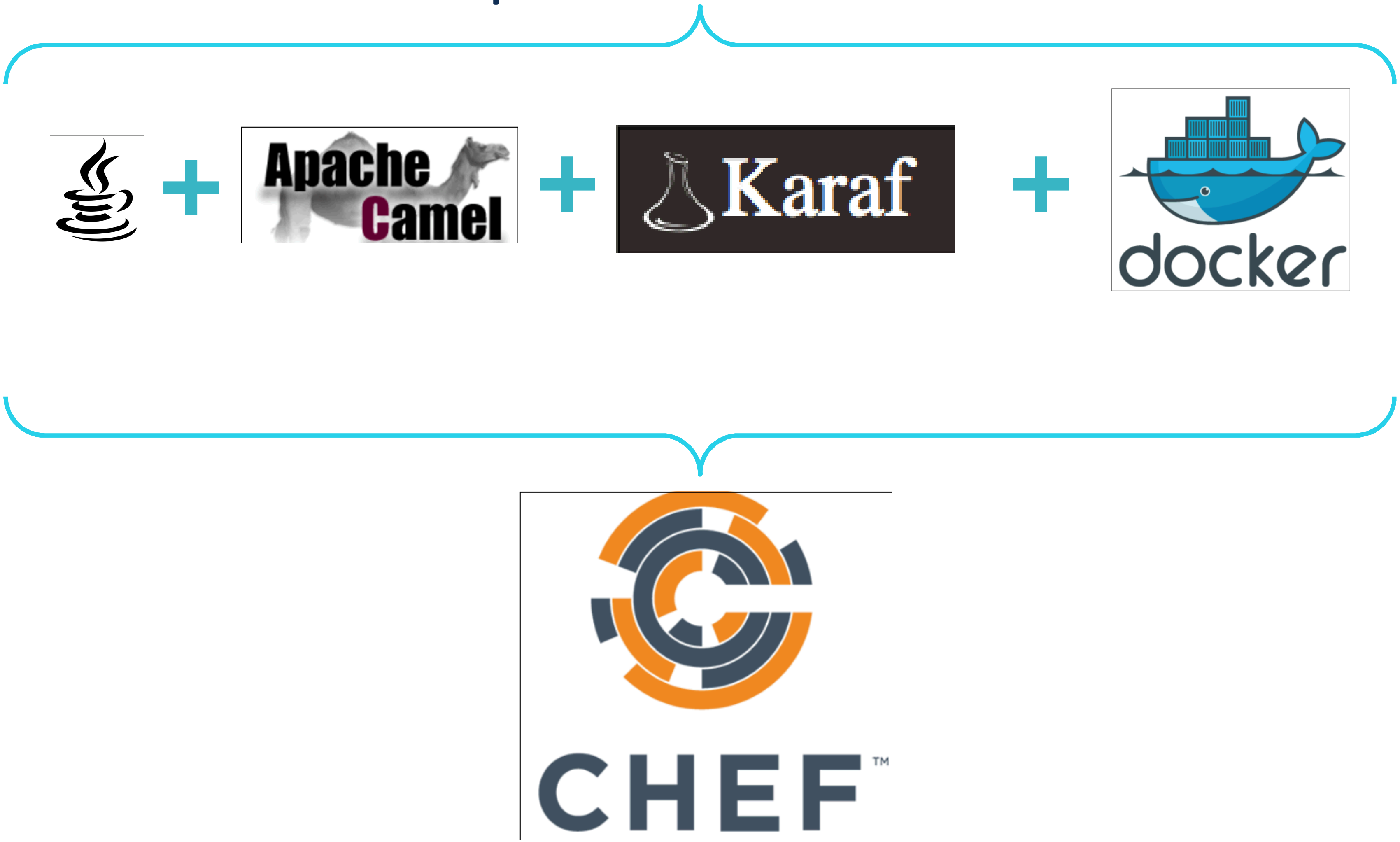
Why ?

- Automation
- Docker single artifact
- Chef ->App->single artifact
- Single Recipe Install
- Chef add registry polling

Artifacts

- Cached Artifacts from Dev->Prod
- Artifacts not stored within Cookbook
- Separates Cookbooks logic from data

Continued Development

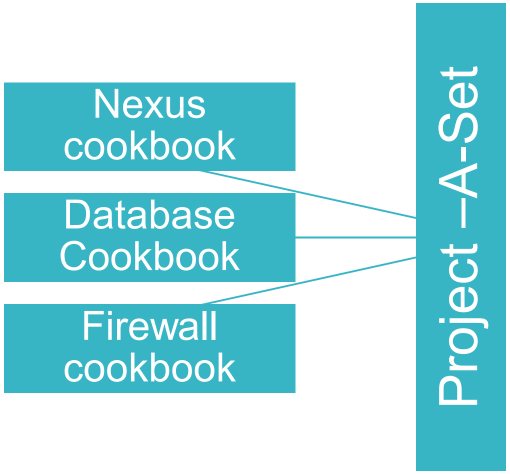


Chef-Countertop Pipeline

- How do we test this?
- Very similar to regular cookbook pipeline
- Stages
  - 1 . Stand up clones in Vsphere per target machine
  - 2. Using chef org in Vsphere (3 from ealier) scp cookbook to workstation
  - 3. From workstation run chef-zero
  - 4. Destroy clones
- Target machines will become a chef production organization
- Machines in our case should not have chef on them
  - Organization-cookbooks should install chef/ then install production cookbooks

Chef-Countertop Pipeline

- Organization cookbook developed last
- All production cookbooks(Role cookbooks) are zipped into Org-cookbooks
  - Therefore they should be done developing



- This cookbook will run from local workstation to standup production environment machines from ground-up (0 -> production ready)

ADVICE FOR NEW DEVELOPERS

..and maybe some current

Virtual Machines

- Attach enough NICS
- Separate NICs per responsibility
  - 1st NIC for Chef connections
  - 2nd NIC for Operational functionality
  - Nth NIC for other management VNC/SSH
- Why?
  - Separate connection downtime
  - If production operation changes IP/Downtime chef will stay up
  - Breakage during operations ->FIX it with chef!
  - Create iptables/routing to ensure strictness

Virtual Machines

- Single NIC
  - Recipe to change IP could break connection to chef server
  - Using Chef-Countertop requires constant connection to workstation
    - Will fail if connection times-out

Pipelines

- Gitlab-runner/Jenkins
- Runner logic wait before Inspec
  - Resetting IP recipes/service
- Transient Machines
  - Much easier to start fresh on new copies than to guess state
  - Script this …obviously......seriously.....
- Containerize software
  - Application layer can be easily deployed with Docker
  - Makes chef scripts simpler separates application settings/system configuration

Recap

- **Architecting The Solution**
  - ***Organization Deployment(Chef-Countertop)***
    - Assume airgapped for also development
    - Countertop contains all dependencies
    - Bootstraps without knife (recipes)
    - Utilizes machine resource “converge_only”
    - Organization cookbook per project
      - Defines your project
      - Traceability of machines
      - Versioning machines
  - ***Cookbook Pipeline***
    - Hard classification of types of cookbooks
    - *Similar process to kitchen*
    - *Large scale*
    - *Chef organization sitting in vsphere*
    - *CI controls all mechanisms*
    - *Cloneable org*
    - *NO ONE UPLOADS TO CHEF-SERVER !!!!!!!!!!*

Closing

- Interesting Techniques
- Chef newbie
- Included as much as I could to help you be successful
  - Pipelines
  - Restricted internet
- Development process is still evolving
  - Learning more everyday
- Glad to be allowed to explore this immense world
  - Very immense

Thank you!