BUILDING DYNAMIC ENVIRONMENTS

Dan Piessens
@dpiessens
ABOUT ME

- Principal Consultant
- 16 Years Experience as Developer, IT Consultant, Architect, Trainer, Coach
- Father of 2.75 kids, T1D and bourbon connoisseur
DANGER: LIVE DEMO

No computers were harmed to favor the demo gods
Monty Python and the Holy Grail

The Holy Book of Days
ACCEPTING THE QUEST
“THE GOAL” FOR DEVOPS

“This book will have a profound effect on IT, just as The Goal did for manufacturing.” --Jez Humble, co-author Continuous Delivery

“This is the IT swamp draining manual for anyone who is neck deep in alligators.” --Adrian Cockroft, Cloud Architect at Netflix

“This is The Goal for our decade, and is for any IT professional who wants their life back.” --Charles Betz, IT architect, author “Architecture and Patterns for IT”
I SEE YOU DON'T USE SOURCE CONTROL

I TOO LIKE TO LIVE DANGEROUSLY
CONTINUOUS INTEGRATION

- Develop
- CI Build
- CI Test
- Package
- Deploy
- Gather Feedback
WHAT MAKES THIS WORK?

- Small Work Batches
- Automated Quality Gates
  - Unit Tests, Code Coverage, Quality Checkers
- Repeatable Process
  - Goal: No Manual Steps During Deployment
PAY DOWN TECH DEBT

TECHNICAL DEBT
EVERYWHERE

TECHNICAL DEBT? YOU KEEP USING THOSE WORDS.
I DO NOT THINK THEY MEAN WHAT YOU THINK THEY MEAN.
OR...

WORKED FINE IN DEV

OPS PROBLEM NOW
THE IDEAL PIPELINE
BUT YOUR APP LOOKS LIKE THIS!
AUTOMATED PROVISIONING LAYERS
INFRASTRUCTURE LAYERING

- **App Platform**
  - Application (Binaries, Configuration)
  - Metadata
    - App Pools, Configuration, Security
  - IIS / Service Hosting

- **Scripted Configuration**
  - Chef, Puppet, PowerShell DSC

- **Virtual Machine Image**
  - HyperV, VMWare, Azure

- **OS Virtualization**
  - HyperV, VMWare, Azure

- **Deploy Pipeline**
  - Fast Moving Dependencies
    - Registry Settings, Tools, Support Files
  - Slow Moving Dependencies
    - Large un-scriptable installs

- **Platform**
**FOUNDATIONS**

- Cloud Encapsulates Environments
  - Allows Quick Creation / Destruction
  - Ensures Isolation
  - Manage Cost

- Azure Resource Management
  - Declarative
  - JSON Based
  - Parameterized and Modular
"resources": [
  {
    "apiVersion": "2015-06-15",
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "IPAddress1",
    "location": "[resourceGroup().location]",
    "tags": {
      "displayName": "IPAddress1"
    },
    "properties": {
      "publicIPAllocationMethod": "Dynamic",
      "dnsSettings": {
        "domainNameLabel": "DNS_NAME"
      }
    }
  },
],
Everything needs to be automated as part of deployment
Create smaller groupings to keep content manageable
Not everything is available at ARM layer!
INSIDE THE VM

- Lots of things get setup after base install
- “Baked” manual images work short term
- But are unmaintainable long term
- Why?
ENTER CHEF

- Modular Infrastructure Configuration — “Recipes”
- Easy DSL
- Dependency Management
- Incorporated Testing
- Huge Community
  - https://supermarket.chef.io
MRP SETUP

# Runs apt-get update
include_recipe 'apt'

# Add the JDK install
include_recipe 'java'

# Install MongoDB
include_recipe 'mongodb'

# Setup Tomcat
mrp_name = 'mrp'
tomcat_install mrp_name

template "/opt/tomcat_#{mrp_name}/conf/server.xml" do
  source 'server.xml.erb'
  owner 'root'
  group 'root'
  mode '0644'
  variables ({{
    :port => "9080",
    :shutdown_port => "8007"
  }})
  notifies :restart, "tomcat_service[#{mrp_name}]"
end

tomcat_service mrp_name do
  action :start
end
APPLICATION DEPLOY

### Dashboard

#### Development
- OctoFX Rate Service: 2.9.6470, July 20th 2015
- OctoFX Trading Website: 2.9.6470, July 20th 2015

#### Test
- OctoFX Rate Service: 2.9.6468, July 20th 2015
- OctoFX Trading Website: 2.9.6470, July 20th 2015

#### Acceptance
- OctoFX Rate Service: 2.9.6398, July 17th 2015
- OctoFX Trading Website: 2.9.6470, July 20th 2015

#### Production
- OctoFX Rate Service: 2.9.6398, July 17th 2015
- OctoFX Trading Website: 2.9.6470, July 20th 2015

### Variables

#### OctoFX Rate Service - Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWSAccessKey</td>
<td>ABCDEFGHIJKLMNOP</td>
<td></td>
</tr>
<tr>
<td>AWSSecretKey</td>
<td>**************</td>
<td></td>
</tr>
<tr>
<td>ServerName</td>
<td>ProdDB01</td>
<td>Production</td>
</tr>
<tr>
<td>ServerName</td>
<td>UATDB01</td>
<td>Acceptance</td>
</tr>
<tr>
<td>ServerName</td>
<td>TBDB01</td>
<td>Test</td>
</tr>
</tbody>
</table>
WHERE SHOULD IT GO?

- Configuration elements can be stored at multiple levels
- Think about shared aspects of the configuration
- Example: DB Connection String
  - Where does it come from?
  - What needs it?
THERE ARE OTHER WAYS!
RELEASE COORDINATION

DO NOT MEDDLE IN THE AFFAIRS OF WIZARDS

for they are subtle and quick to anger.
Deployments should be granular
- Easier to maintain, template and update

Application = Many Deployments
- Services
- Websites
- Databases
- Infrastructure

Coordination can be a pain!
PU DEPLOYMENT

1. ARM Template VM
2. Chef Configure VM
3. Register In Octopus
4. Deploy Applications
5. ARM Template Site
6. Deploy Database
7. Deploy Website
SO MANY TOOLS!

- Chef
- Azure DevOps
- Docker
- Octopus Deploy
RELEASE ORCHESTRATION

- Want the one-click deployment back
- Needs to coordinate all layers
- Ensures no missing dependencies
- Should work for every environment including Production
SUPPORTING TOOLS

- Chef Delivery
- Microsoft Release Management
- IBM UrbanCode Release
- XebiaLabs XL Release
- Octopus Deploy 3.4
CARE AND FEEDING

- This will require **constant** maintenance!
- Pro Tip: Add these items to your DoD
- Everyone needs to understand and troubleshoot the pipeline
THE TESTING PYRAMID

UI (1-5%)

Service Tests (5-15%)

Unit Tests (80-94%)

Mike Cohn, *Succeeding with Agile*
TEST PYRAMID EXAMPLE TOOLS

UI
Selenium, Coded UI

Service, Integration & Acceptance
Unit Testing Frameworks

Unit
JUnit, MSTest, Jasmine, XUnit

BDD frameworks (Cucumber, SpecFlow)
INFRASTRUCTURE TWIST

- **Ui’ish**
  - InSpec, Azure Load Testing

- **Integration & Acceptance**
  - Unit Testing Frameworks

- **Unit**
  - Pester, Rspec, ChefSpec
REMEMBER THE BASICS

- Unit tests should run isolated at build time
- Integration tests should be API level
- Minimize “UI” tests to key scenarios
Describe "BuildIfChanged" {
  Context "When there are changes" {
    Mock Get-Version {return 1.1}
    Mock Get-NextVersion {return 1.2}
    Mock Build {} -Verifiable -ParameterFilter { $version -eq 1.2 }

    $result = BuildIfChanged

    It "Builds the next version" { Assert-VerifiableMocks }
    It "Returns the next version number" { $result | Should Be 1.2 }
  }
}
describe 'directory::create' do

  let(:chef_run) {
    ChefSpec::SoloRunner.converge(described_recipe)
  }

  it 'creates a directory with the default action' do
    expect(chef_run).to create_directory('/tmp/default_action')
    expect(chef_run).to_not create_directory('/tmp/not_default_action')
  end
end
control 'sshd-8' do
  impact 0.6
  title 'Server: Configure the service port'
  desc '
     Always specify which port the SSH server should listen to. Prevent unexpected settings.
       
       tag "tag_name"
       ref "name", url: "https://..."
  describe sshd_config do
    its('Port') { should eq('22') }
  end
end
INTEGRATION TESTS

- Author in frameworks that report well
- Let the tools that run it best do the running
- Manage data as part of deployment
- Have the deployment tool supervise
GETTING FEEDBACK
RUNTIME FEEDBACK

- Instrument your applications at runtime!
- Many tools available
  - Application Insights
  - New Relic
  - Splunk
  - Raygun.io
- Include User Analytics
DIG DEEPER FOR DATA

- Monitoring tools have APIs for tracing / logging
- Instrument key transactions in your system
- Track performance end-over-end for deployments
- Choose a logging framework and leverage it!
  - Log to a common location
LOAD TESTING

- DON’T DO IT
  - Users are weird… we can’t emulate them

- Take performance baselines

- Compare new builds with baseline
FREEZING FAILURES

- Tennant model allows you to snapshot an environment
- Determine how long to keep them
- Make failures visible
- Remember: inactive does not mean free in the cloud!
TAKE AWAY CONCEPTS

- Understand the basics before undertaking
- Modularize applications, understand dependencies
- Orchestrate all release aspects
- Performance baselines only
- Test your infrastructure!

“Deployments are like exercise, the more you do them, the less it hurts”

-Dan Piessens
RESOURCES

- **Azure Resource Manager**

- **Chef**
  - http://chef.io

- **Octopus Deploy**
  - http://www.octopusdeploy.com

- **Today’s Code**
  - https://github.com/dpiessens/PartsUnlimitedDemo
THANK YOU!

QUESTIONS?