title {
  name = "the enterprise defined as code"
}

11/9/19
about {
  name = anthony rees
  role = sa apac
  company = chef software
  twitter = @anthonyrees
}
What is this talk really about?

title {
    name = "the coded enterprise"
    subject = "devops"
}
question {
    this = "why do we have the stack?"
}
The Enterprise “Stack” as code

Why manage the stack as code?

- Can manage versions of code and evolve it easily
- Can test code to be sure it works and inspect it for accuracy and safety
- Code is unambiguous - it does what is expected
- Can use established software development lifecycle to manage code
Why do we have the “Stack”?

Because the application provides business value!
Why do we have the “Stack”?

Supporting the application is managing its RISK.
section {
    name = "the stack as code"
}
Application Stack as Code

- Application
- Configuration
- Provisioning
- Operating System
- Compliance
- Patch Security
remember { 
  this = "use the right tool for the job!"
}

11/9/19
Application Stack as Code

InSpec – Ruby

```ruby
control "xccdf_org.cisecurity.benchmarks_rule_1.1.3_L1_Ensure_Minimum_password_age" do
  title "(L1) Ensure 'Minimum password age' is set to '1 or more day(s)'."
  desc "This policy setting determines the number of days that you must wait before changing a user password.

  The recommended state for this setting is: 1 or more day(s).

  Rationale: Users may have favorite passwords that they like to use.

  Impact: 1.0"
  describe security_policy do
    its("MinimumPasswordAge") { should be => 1 }
  end
end
```
Application Stack as Code

Packer – JSON

```json
{
  "region": "us-east-1",
  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "{{ user \"aws_access_key\" }}",
      "secret_key": "{{ user \"aws_secret_key\" }}",
      "region": "{{ user \"region\" }}",
      "instance_type": "t2.micro",
      "source_ami_filter": {
        "filters": {
          "virtualization-type": "hvm",
          "name": "*Windows_Server-2012-R2*English-64Bit-Base*",
          "root-device-type": "ebs"
        },
        "most_recent": true,
        "owners": "amazon"
      },
      "ami_name": "packer-demo-{{timestamp}}"
    }
  ]
}
```
Application Stack as Code

Terraform | ARM | CloudFormation

```hcl
resource "aws_internet_gateway" "national_parks_gateway" {
  vpc_id = "${aws_vpc.national_parks_vpc.id}"

  tags {
    Name = "${var.tag_name}_national_parks_gateway-${var.tag_application}"
  }
}

resource "aws_route" "national_parks_internet_access" {
  route_table_id = "${aws_vpc.national_parks_vpc.main_route_table}"
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = "${aws_internet_gateway.national_parks_gateway}"
}

resource "aws_subnet" "national_parks_subnet" {
  vpc_id = "${aws_vpc.national_parks_vpc.id}"
}
```

APPLICATION
CONFIGURATION
PROVISIONING
OPERATING SYSTEM
COMPLIANCE PATCH SECURITY
Application Stack as Code

Chef – Ruby

```ruby
package 'httpd' do
  action :install
end

template '/var/www/html/index.html' do
  source 'index.html.erb'
end

template '/etc/httpd/conf/httpd.conf' do
  source 'httpd.conf.erb'
  notifies :restart, 'service[httpd]' end

service 'httpd' do
  action [:enable, :start]
end
```
Application Stack as Code

Habitat – Shell | Powershell

```powershell
$pkg_name="sqlwebadmin"
$pkg_origin="mwrock"
$pkg_version="0.1.0"
$pkg_maintainer="Matt Wrock"
$pkg_license=@('MS-PL')
$pkg_description="Web based SQL Server Administrator"
$pkg_deps=@("core/dsc-core", "core/sql-dmo", "core/iis-webserverrole"
$pkg_source="https://codeplexarchive.blob.core.windows.net/archive/p
$pkg_sha256="ea888d26a989951a62e5ac60f4f819ee0662f5e6d99edeb4896a628"
$pkg_upstream_url="https://archive.codeplex.com/?p=sqlwebadmin"

$pkg binds optional=@{
    "database":"instance username password port"
}

function Invoke-Unpack {
    Invoke-DefaultUnpack
```
## Tool -> Code -> Build -> Artefact

<table>
<thead>
<tr>
<th>Full Stack</th>
<th>Tool</th>
<th>Code</th>
<th>Build</th>
<th>Deliver</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td>Habitat</td>
<td>plan.sh/ps1</td>
<td>.hart</td>
<td>Habitat Depot</td>
</tr>
<tr>
<td>Config</td>
<td>Chef</td>
<td>recipe.rb</td>
<td>Cookbuild</td>
<td>Chef Server</td>
</tr>
<tr>
<td>Provision</td>
<td>Terraform</td>
<td>main.tf</td>
<td>.tfstate</td>
<td>Virtual Network, Compute, Storage</td>
</tr>
<tr>
<td></td>
<td>CloudFormation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Packer</td>
<td>os.json</td>
<td>iso / pac jr iso upload</td>
<td>AWS Azure vm / VMDK</td>
</tr>
<tr>
<td>Compliance</td>
<td>inspec</td>
<td>Inspec.rb</td>
<td>profile</td>
<td>A2</td>
</tr>
</tbody>
</table>
section {
    name = "one way to production"
}
Package any Windows or Linux application with Habitat

Local Development

Source Control

CI/CD

Artifact Store

Environments

kubernetes

Azure

aws

vmware

acceptance

staging

production
Check feature branch into git based source control tools like GitHub Enterprise, GitLab, & Stash
Verify your commits with automated syntax, lint, and unit tests.
Review changes with peers before merging commits to master, and gain visibility into change approval process.

- Source Control
  - GitLab
  - Bitbucket

- CI/CD

- Artifact Store

- Environments
  - Acceptance
  - Staging
  - Production
Use existing CI/CD tooling already deployed within your environment.
Run automated functional and integration tests before delivering your application.
remember {
    this = “containers don’t change your approach!”
}

Deliver your Habitat packaged application to the Habitat Depot, or export to container registries like Docker Hub.
section {
    name = "compliance at code"
}

### 1.1.10 Add nodev Option to /home (Scored)

**Profile Applicability:**
- Level 1

**Description:**
When set on a file system, this option prevents character and block special devices from being defined, or if they exist, from being used as character and block special devices.

**Rationale:**
Since the user partitions are not intended to support devices, set this option to ensure that users cannot attempt to create block or character special devices.

**Note:** The actions in the item refer to the `/home` partition, which is the default user partition that is defined in CentOS 6. If you have created other user partitions, it is recommended that the Remediation and Audit steps be applied to these partitions as well.

**Audit:**
Run the following commands to determine if the system is configured as recommended.

```
# grep /tmp /etc/fstab | grep nodev
# mount | grep /tmp | grep nodev
```

If either command emits no output then the system is not configured as recommended.

**Remediation:**
Edit the `/etc/fstab` file and add `nodev` to the fourth field (mounting options). See the `fstab(5)` manual page for more information.

```
# mount -o remount,nodev /home
```

### 1.1.11 Add nodev Option to Removable Media Partitions (Not Scored)

**Profile Applicability:**

**Description:**
Set `nodev` on removable media to prevent character and block special devices that are present on the removable be treated as these device files.

**Rationale:**
Removable media containing character and block special devices could be used to circumvent security controls by allowing non-root users to access sensitive device files such as `/dev/kmem` or the raw disk partitions.

**Audit:**

```
# grep <each removable media mountpoint> /etc/fstab
Verify that nodev is an option
```

**Remediation:**
Edit the `/etc/fstab` file and add "nodev" to the fourth field (mounting options). Look for entries that have mount points that contain words such as floppy or cdrom. See the `fstab(5)` manual page for more information.

### 1.1.12 Add noexec Option to Removable Media Partitions (Not Scored)

**Profile Applicability:**
- Level 1

**Description:**
Set `noexec` on removable media to prevent programs from executing from the removable media.

**Rationale:**
Setting this option on a file system prevents users from executing programs from the removable. This deters users from being to introduce potentially malicious software on the system.

**Audit:**

```
# grep <each removable media mountpoint> /etc/fstab
Note: Verify that noexec is an option
```
Product Ideas and Features

Security Review
We have a Communication Problem
Application Stack as Code

InSpec – Ruby

```ruby
control "xccdf_org.cisecurity.benchmarks_rule_1.1.3_L1_Ensure_Minimum_password_age" do
  desc "This policy setting determines the number of days that you must wait before changing your password.

  The recommended state for this setting is: 1 or more day(s).

  Rationale: Users may have favorite passwords that they like to use.
  
  impact 1.0
  
  describe security_policy do
    its("MinimumPasswordAge") { should be >= 1 }
  end
end
```
How does the code look?

describe service('apache2') do
  it {} { should be_running }
end

describe port(80) do
  it {} { should be_listening }
end

describe http('http://localhost', enable_remote_worker: true) do
  its('status') { should cmp 200 }
  its('body') { should match /Welcome to / }
end
section {
    name = "application automation"
}
Habitat enables application teams to build, deploy, and manage any application in any environment - from traditional data centers to containerized microservices.

1. “Lift & Shift” Legacy Apps to Modern Platforms
   Organizations struggle to move existing, business critical apps to modern platforms

2. Deliver on a Cloud-Native (Cloud/Containers) Strategy
   Organizations hit a wall when adopting and deploying to a cloud-native platform
How does it work?

It splits the platform-independent part of the application from the platform-dependent part.
How does it work?

- All of the problems shown previously are a result of this pattern: building up from the operating system.
- The entire triangle becomes the artifact you carry around with you now and in the future (including sometimes the VM and the server!)

- Habitat builds from the application down
- Embedded supervisor as management interface
- Builds have strict dependency version control
Building Applications with Habitat

Packaging Applications

Running Applications
Deploying & Managing Apps with Habitat

Running Applications

Topologies

Update Strategies

- LEADER
- STAND ALONE
- LEADER ELECTION
- ALL AT ONCE
- ROLLING
section {
    name = "the wrap up"
}
DEVOPS TALKS CONFERENCE 2019

**APPLICATION**

**COMPLIANCE**

**INFRASTRUCTURE**

- Source Control
- CI/CD
- Artifact Store
- Environments

- Acceptance
- Staging
- Production

- GitLab
- Bitbucket

- Acceptance
- Staging
- Production

- CI/CD

- Acceptance
- Staging
- Production

- Artifact Store

- Acceptance
- Staging
- Production

- Environments

11/9/19
competition {
    win = "nintendo"
    contact = "chef"
}
section {
  name = "thank you"
  contact = "anthony@chef.io"
  twitter = "@anthonyrees"
}