# Advanced Dungeons and Testing Dragons

Temple of the Golden Cobra

**SLIDES: https://goo.gl/56EJwk**





CHEF

# Franklin Webber

*Training and Technical Content Lead*

US Army - Radio/COMSEC Repairer (35E)

Tech Support

Quality Assurance

Software Developer Eng. in Test

Software Developer

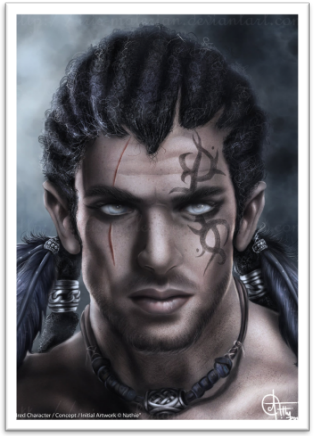
Instructor / Trainer





CHEF

**Dan**
**"Two Swords"**
**Robinson**



**Eric**
***"BB / LG"***
**Maxwell**

**Kimball**
**"Magic Missile"**
**Johnson**


**Tyler**
***"Cure Light Wounds"***
**Fitch**





CHEF

# You

*Level 4-6 Chef Cookbook Author*

High Charisma

Understand Chef Resources, Cookbooks, and Cookbook dependencies

Understanding testing basics (i.e. ChefSpec and the RSpec command)





CHEF

# Introduce Yourself

- Name
- Where you work
- The Best Part of Your Job
- The Most Difficult Part of Your Job
- A Favorite Hobby or Past time as a Child
- What You Want to get out of ChefConf

# Schedule

```
 9:00 -  9:30 - Introductions
 9:30 - 10:20 - Finding the Ark
10:20 - 10:30 - BREAK
10:30 - 11:20 - Unraveling the Power of the Ark
11:20 - 11:30 - BREAK
11:30 - 12:20 - The Ark's Power Realized
12:20 - 13:45 - LUNCH
```



CHEF

# Schedule

12:20 - 13:45 - LUNCH

13:45 - 14:35 - The Python's Scales

14:35 - 14:45 - BREAK

14:45 - 15:35 - Challenging the Golden Cobra

15:35 - 15:45 - BREAK

15:45 - 16:35 - Securing the Power of the Ark

16:35 - 16:45 - GOODBYES





CHEF

# Scenario Agenda

- ❑ **Introduction** (3 minutes)

  D&D flavored introduction

- ❑ **The Encounter** (8 minutes)

  Outline the problem that you are going to face

- ❑ **Concepts** (10 minutes)

  Define the tools and techniques you are going to use to solve the problem

- ❑ **Exercise** (20 minutes)

  Give you time to solve the problem with the support of us and the party

- ❑ **Review** (10 minutes)

  Demonstration and Review the techniques and answer questions

# Advanced Dungeons and Testing Dragons

Temple of the Golden Cobra

**SLIDES: https://goo.gl/56EJwk**





CHEF

# Finding the Ark

- Introduction
- The Encounter
- Concepts
- Exercise
- Review

# Finding the Ark

- ✓ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

# Changing into the Cookbook Directory

```
> cd ~/ark
```





CHEF

# Executing the Test Suite

```
> chef exec rspec spec/recipes/default_spec.rb
```

```
..................

Finished in 3.58 seconds (files took 2.73 seconds to load)
19 examples, 0 failures



ChefSpec Coverage report generated...

  Total Resources:   41
  Touched Resources: 41
  Touch Coverage:    100.0%
```

# Viewing the Recipe Specification

~/ark/spec/recipes/default_spec.rb

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on an unspe...orm' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end

    it 'installs necessary packages' do
```

MOTIVATION

# Redundancy in the Test Suite

As a test suite grows in size and complexity it becomes more difficult to read and maintain.

This will ultimately make it more difficult to make changes to the cookbook as you will often times fight with the specifications more often than writing the code that will deliver value to the goals of our organization.

**Adventurers, your goal is to reduce the size and complexity of this test file.**

# Finding the Ark

- ✓ Introduction
- ✓ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

CONCEPT

# Power Word: let

Use let to define a memoized helper method. The value will be cached across multiple calls in the same example but not across examples.

Note that let is lazy-evaluated: it is not evaluated until the first time the method it defines is invoked. You can use let! to force the method's invocation before each example.

https://goo.gl/BJp0IQ





CHEF

# Diagramming the let Helper Method

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'instal necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. let is a RSpec helper method
2. Ruby Symbol
3. Code Block
4. Invocation

# Invoking the Helper Method

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'installs necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. chef_run sends a message
2. RSpec invokes the contents of the block
3. RSpec stores the contents of the execution
4. chef_run sends a message
5. RSpec retrieves the stored execution

# Cached Within each Example

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'installs necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. chef_run is loaded and stored
2. chef_run uses the stored invocation
3. chef_run is loaded and stored

# A chef_run with Node Attributes

```
describe 'ark::default' do
  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new({ platform: 'centos',
                                          version: '6.7' })
      runner.converge(described_recipe)
    end

    # ... EXAMPLES WITHIN CONTEXT
  end
end
```

# Using let to Create Clearer Examples

```
describe 'ark::default' do
  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new(node_attributes)
      runner.converge(described_recipe)
    end

    let(:node_attributes) do
      { platform: 'centos', version: '6.7' }
    end

    # ... EXAMPLES WITHIN CONTEXT
  end
```

# A let for each Context

```
describe 'ark::default' do
  context 'when no attributes are specified, on an unspecifed ...orm' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end
    # ... EXAMPLES WITHIN CONTEXT
  end

  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new(platform: 'centos', version: ...)
      runner.converge(described_recipe)
    end
    # ... EXAMPLES WITHIN CONTEXT
```

# Overriding a let in a Child Context

```
describe 'ark::default' do
  let(:chef_run) do
    runner = ChefSpec::SoloRunner.new(node_attributes)
    runner.converge(described_recipe)
  end

  let(:node_attributes) do
    {}
  end

  context 'when no attributes are specified, on an unspecif...orm' do
    # ... EXAMPLES WITHIN CONTEXT
  end

  context 'when no attributes are specified, on CentOS' do
    let(:node_attributes) do
      { platform: 'centos', version: '6.7' }
    end
    # ... EXAMPLES WITHIN CONTEXT
```

CONCEPT

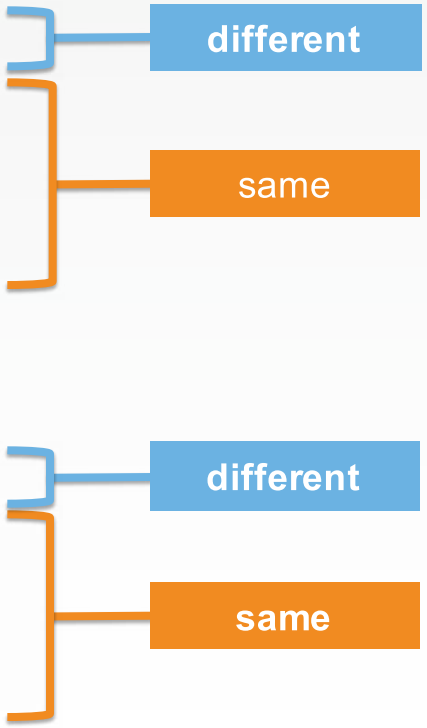
# Conjure shared_examples

Shared examples let you describe behavior of classes or modules. When declared, a shared group's content is stored. It is only realized in the context of another example group, which provides any context the shared group needs to run.

https://goo.gl/yi12tM

# Finding Similar Expressed Expectations

```
context 'when no attributes are specified, on an unspecif...orm' do
  # let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
context 'when no attributes are specified, on CentOS' do
  # let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
```






CHEF

# A Place to Share Examples

```
shared_examples 'installs packages' do
  it 'installs the necessary packages' do
    packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
context 'when no attributes are specified, on an unspecif...orm' do
  # let(:installed_packages) ...
  it_behaves_like 'installs packages'
end
context 'when no attributes are specified, on CentOS' do
  # let(:installed_packages) ...
  it_behaves_like 'installs packages'
end
```

# Finding the Ark

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ❑ Exercise
- ❑ Review

ENCOUNTER

# Exercise

- Refactor the the Ark's default recipe specification using:

- **Power Word: let**
- **Conjure: shared_examples**

- Run `chef exec rspec spec/recipes/default_spec.rb` to ensure the examples pass

# Success

All tests pass and the size of the recipe specification has been decreased.


Finding the Ark
✓ Introduction
✓ The Encounter
✓ Concepts
✓ Exercise
❑ Review

# Change in to the Cookbook Directory

```
> cd ~/ark
```





CHEF

# Execute the Test Suite

```
> chef exec rspec spec/recipes/default_spec.rb
```

```
..................

Finished in 3.58 seconds (files took 2.73 seconds to load)
19 examples, 0 failures



ChefSpec Coverage report generated...

  Total Resources:   41
  Touched Resources: 41
  Touch Coverage:    100.0%
```

# Edit the Recipe Specification File

~/ark/spec/recipes/default_spec.rb

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on an ...platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end

    it 'installs necessary packages' do
```

DEMO

# Live Demonstration

**https://goo.gl/ChkP47**





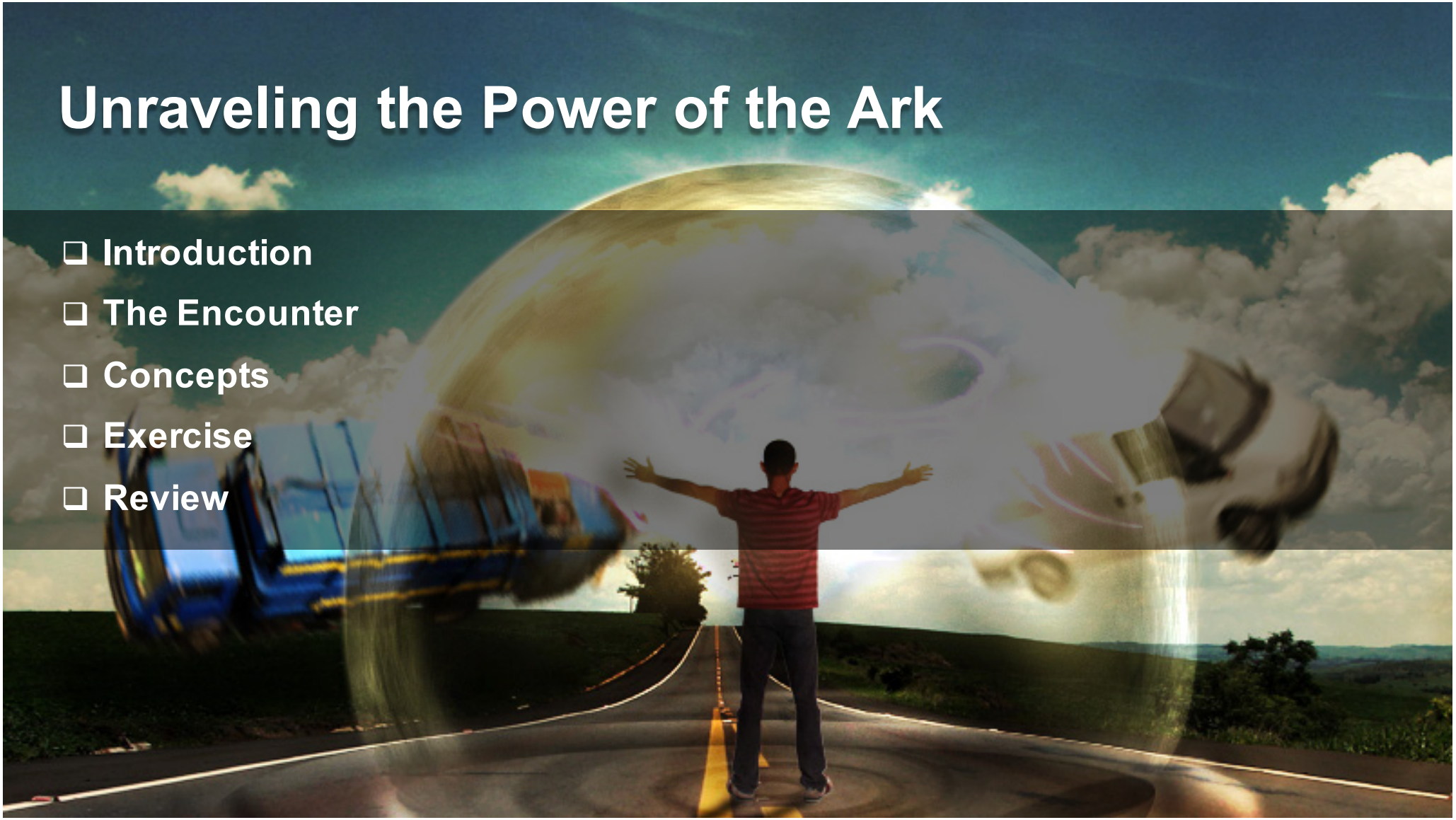
CHEF

# Finding the Ark

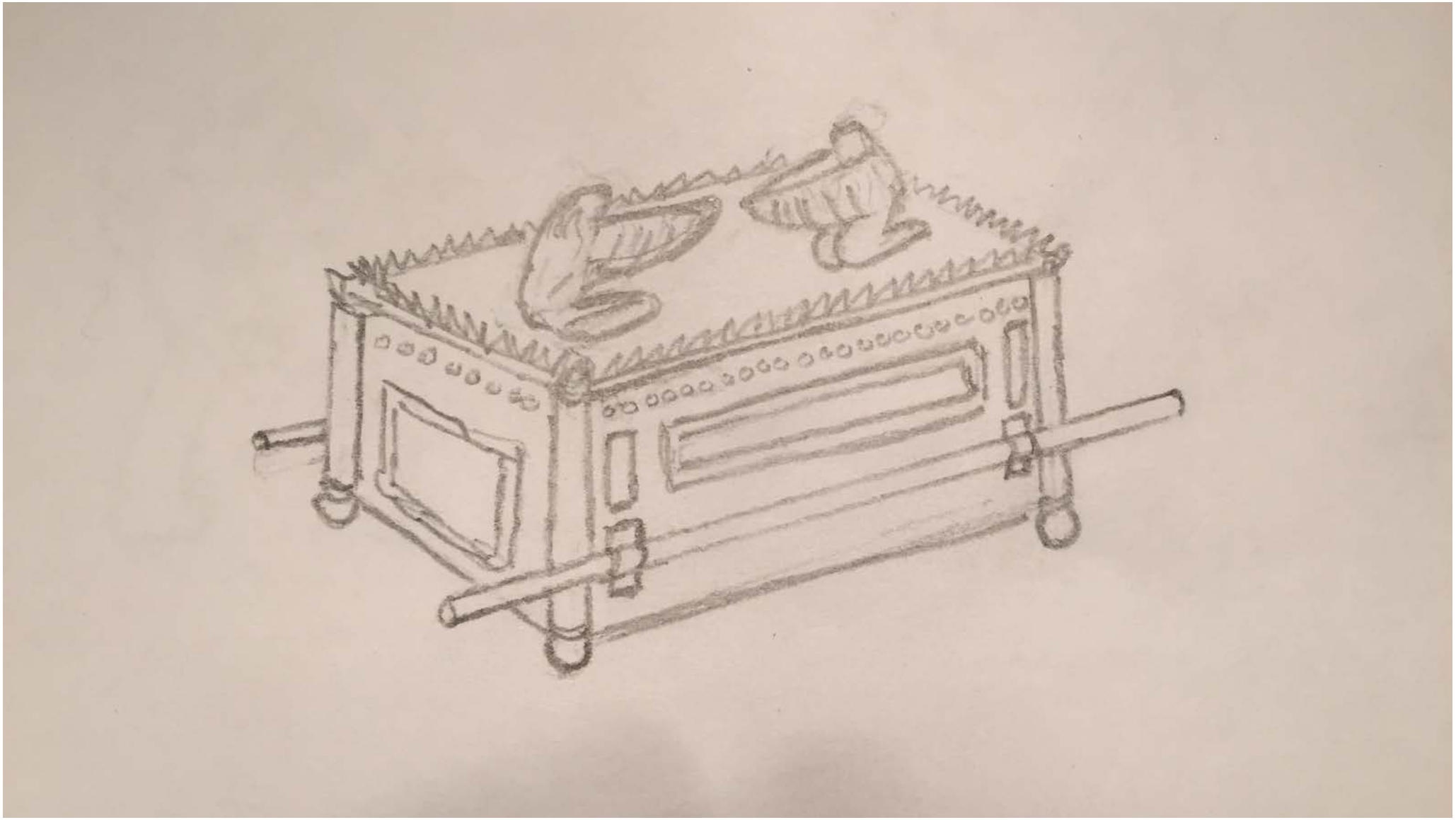
- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ✓ Review

CHEF™

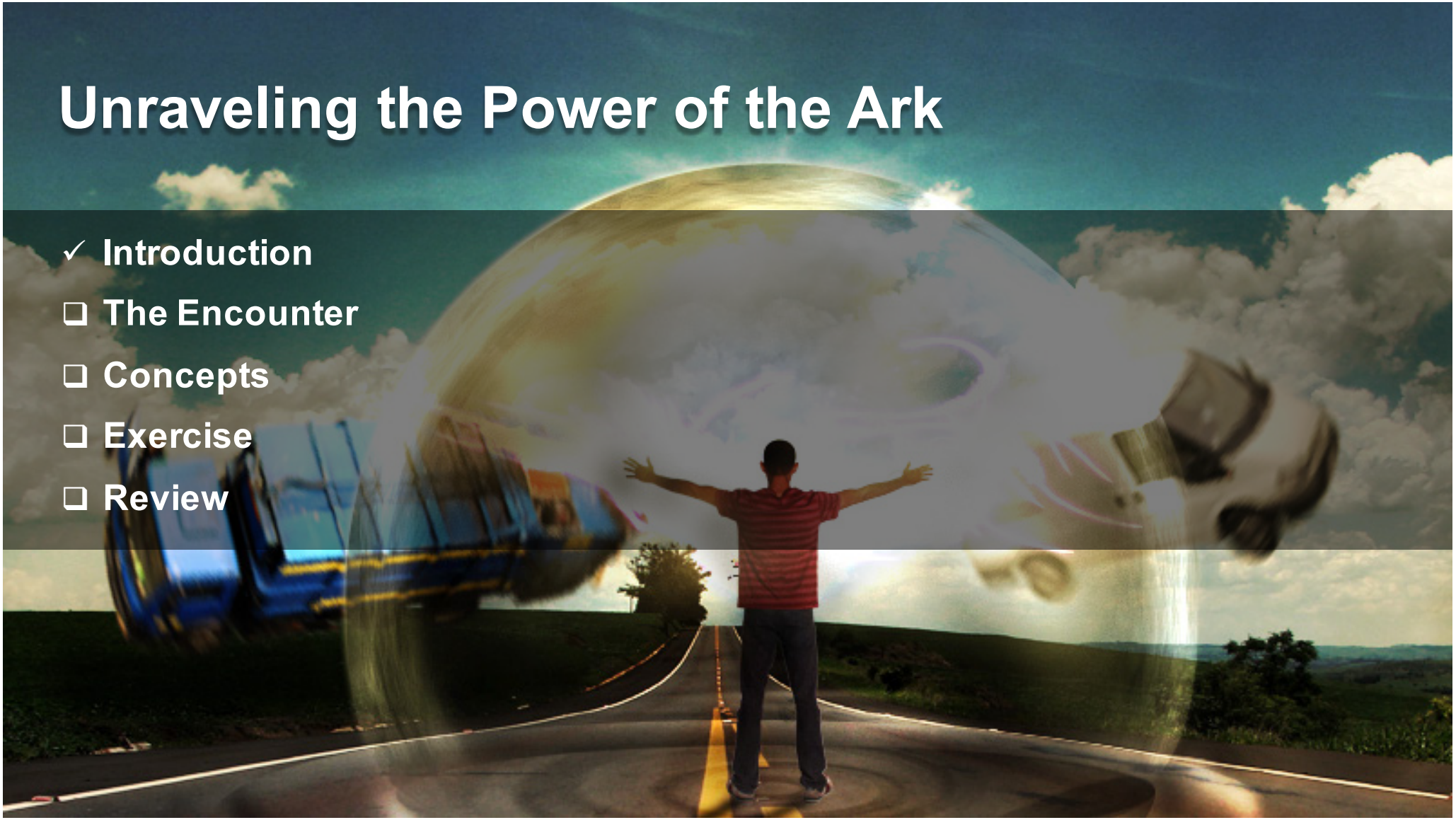
# Unraveling the Power of the Ark



- Introduction
- The Encounter
- Concepts
- Exercise
- Review



# Unraveling the Power of the Ark



- ✓ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

# Changing into the Cookbook Directory

```
> cd ~/ark
```





CHEF

# Executing the Test Suite

```
> chef exec rspec spec/recipes/default_spec.rb
```

```
..................

Finished in 3.58 seconds (files took 2.73 seconds to load)
19 examples, 0 failures



ChefSpec Coverage report generated...

  Total Resources:   41
  Touched Resources: 41
  Touch Coverage:    100.0%
```





CHEF

# Viewing the Recipe Specification

~/ark/spec/recipes/default_spec.rb

```
require 'spec_helper'

describe 'ark::default' do
  let(:chef_run) do
    runner = ChefSpec::SoloRunner.new(node_attributes)
    runner.converge(described_recipe)
  end

  let(:node_attributes) do
    {}
  end
```

MOTIVATION

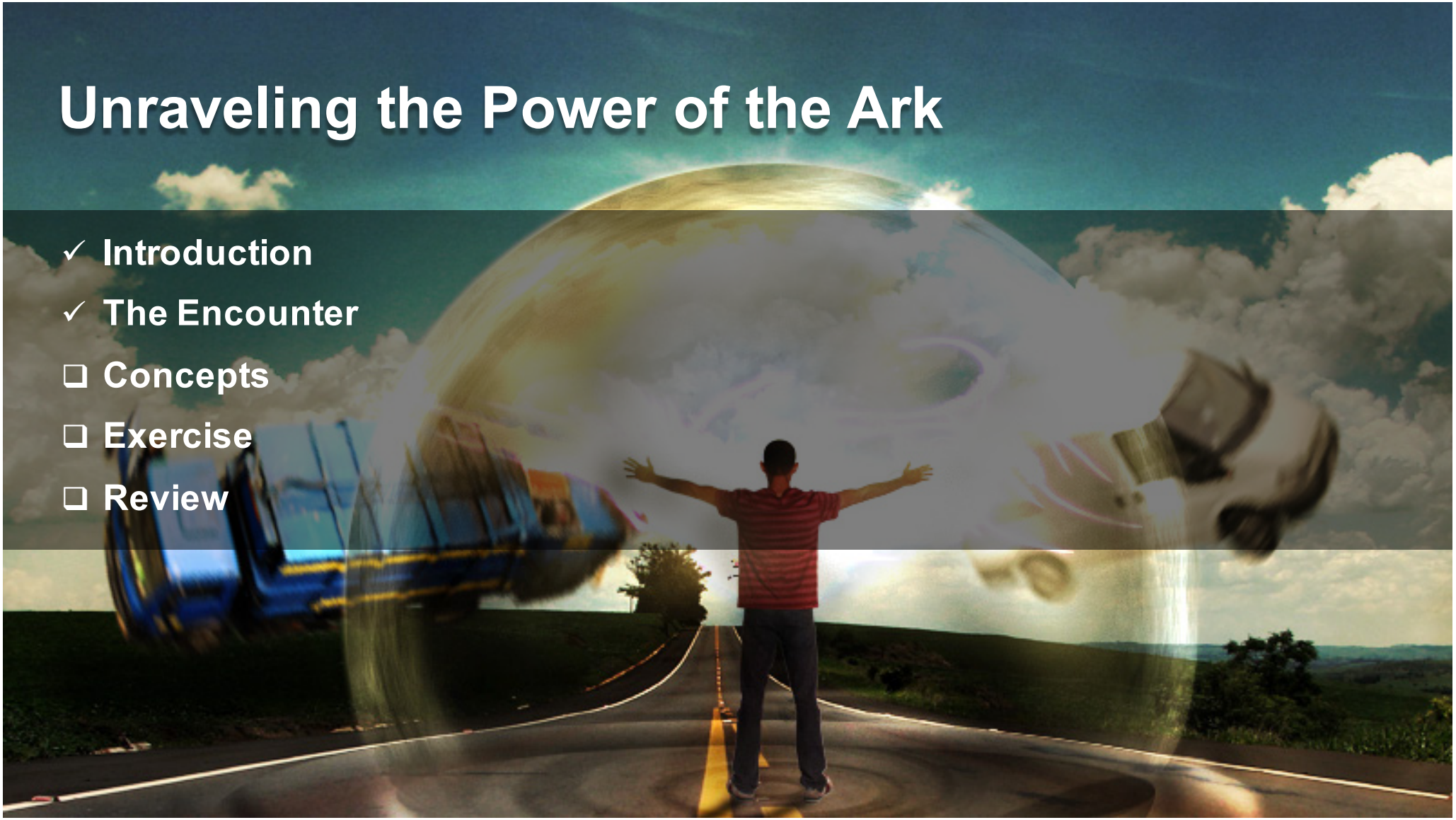
# Reusing More of the Test Suite*

The solutions you found (or started to find) would likely be useable by other tests with the same cookbook and possibly another cookbook.

There are a few more spells and abilities one could apply to help combat the redundancy in this test file.

**Adventurers, your goal is to further reduce the size and complexity of this test file.**

# Unraveling the Power of the Ark



- ✓ Introduction
- ✓ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

CONCEPT

# Prismatic require

Loads the given name, returning true if successful and false if the feature is already loaded.

If the filename does not resolve to an absolute path, it will be searched for in the directories listed in $LOAD_PATH ($:).

http://goo.gl/cLKY37

# Requiring the spec_helper file

~/ark/spec/recipes/default_spec.rb

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on an ...platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end
```

# Viewing the spec_helper file

~/ark/spec/spec_helper.rb

```
require 'chefspec'
require 'chefspec/berkshelf'

at_exit { ChefSpec::Coverage.report! }

RSpec.configure do |config|
  config.color = true
end

# puts $LOAD_PATH
# ... define test helpers and content in this file
```

# Viewing the $LOAD_PATH

```
> chef exec rspec
```

```
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/uuidtools-2.1.5/lib
/Users/franklinwebber/ark/spec
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-support-3.4.1/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-core-3.4.4/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/net-ssh-3.1.1/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/fauxhai-3.5.0/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/diff-lcs-1.2.5/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-expectations-3.4.0/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-mocks-3.4.1/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-3.4.0/lib
```

CONCEPT

# Greater require_relative

Ruby tries to load the library named string relative to the requiring file's path. If the file's path cannot be determined a LoadError is raised. If a file is loaded true is returned and false otherwise.

http://goo.gl/Gv8QdI

CONCEPT

# Create helper method

You can define a Ruby method that takes care of some of the tedious work of retrieving node attributes.





CHEF

# Viewing the Repetition of Retrieving Attributes

```
it "apache mirror" do
  attribute = chef_run.node['ark']['apache_mirror']
  expect(attribute).to eq "http://apache.mirrors.tds.net"
end

it "prefix root" do
  attribute = chef_run.node['ark']['prefix_root']
  expect(attribute).to eq "/usr/local"
end
```

# Refactoring with let to help ease the pain

```
let(:node) do
  chef_run.node
end

it "apache mirror" do
  attribute = node['ark']['apache_mirror']
  expect(attribute).to eq "http://apache.mirrors.tds.net"
end

it "prefix root" do
  attribute = node['ark']['prefix_root']
  expect(attribute).to eq "/usr/local"
end
```

# Implementing a Helper Method

```
let(:node) do
  chef_run.node
end

def attribute(name)
  node[described_cookbook][name]
end
it "apache mirror" do
  expect(attribute('apache_mirror')).to eq "http://apache.mirr....net"
end

it "prefix root" do
  expect(attribute('prefix_root')).to eq "/usr/local"
end
```

CONCEPT

# Ray of shared_context

Use shared_context to define a block that will be evaluated in the context of example groups either explicitly, using include_context, or implicitly by matching metdata.

http://goo.gl/R0ujTA

# Repeating More than Examples

```
describe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new(node_attributes)
      runner.converge(described_recipe)
    end

    let(:node_attributes) do
      {}
    end

    def attribute(name) ...
```

# Repeating More than Examples

```
shared_context 'converged recipe' do
 let(:chef_run) do
   runner = ChefSpec::SoloRunner.new(node_attributes)
   runner.converge(described_recipe)
 end

  let(:node_attributes) do
    {}
  end

  def attribute(name) ...
end

describe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    include_context 'converged recipe'
```

CONCEPT

# alias_example_group_to

**describe** and **context** are the default aliases for **example_group**. You can define your own aliases for **example_group** and give those custom aliases default metadata.

http://goo.gl/DfUChL

# Viewing a Spec with include_context

~/ark/spec/recipes/default_spec.rb

```
describe 'ark::default' do
  context 'when no attributes are specified, on an ...platform' do
    include_context 'converged recipe'

    # ... examples ...

  end

  # ... other contexts ...
end
```

# Refactoring the Spec to auto include the context

```
describe_recipe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    include_context 'converged recipe'

    # ... examples ...

  end

  # ... other contexts
end
```

# Defining the alias and tying it to the shared_context

~/ark/spec/spec_helper.rb

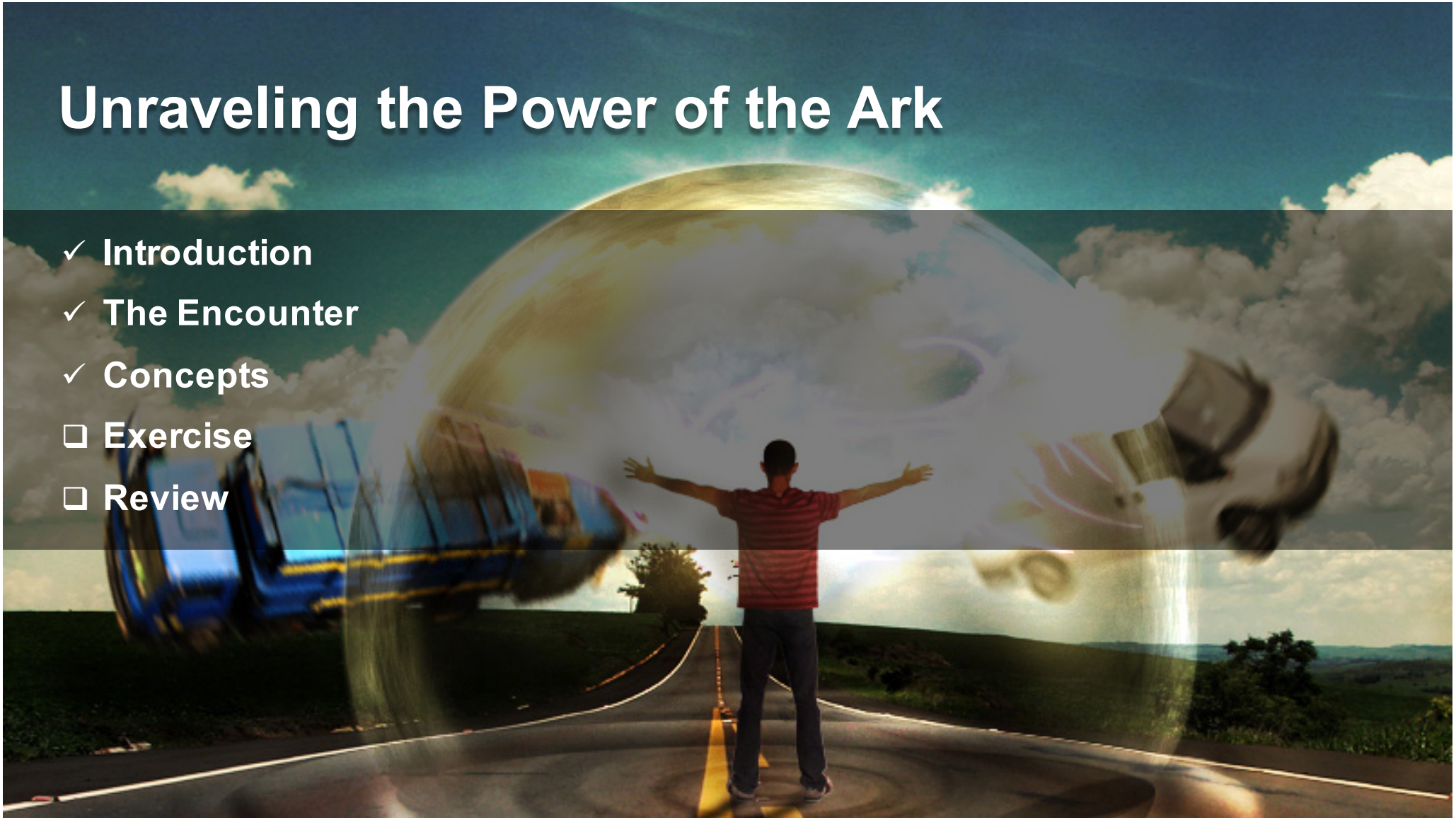
```
require 'chefspec'
require 'chefspec/berkshelf'

at_exit { ChefSpec::Coverage.report! }

RSpec.configure do |config|
  config.color = true
  config.alias_example_group_to :describe_recipe, :type => :recipe
end

shared_context 'converged recipe', :type => :recipe do
```

# Unraveling the Power of the Ark



- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ❑ Exercise
- ❑ Review

ENCOUNTER

# Exercise

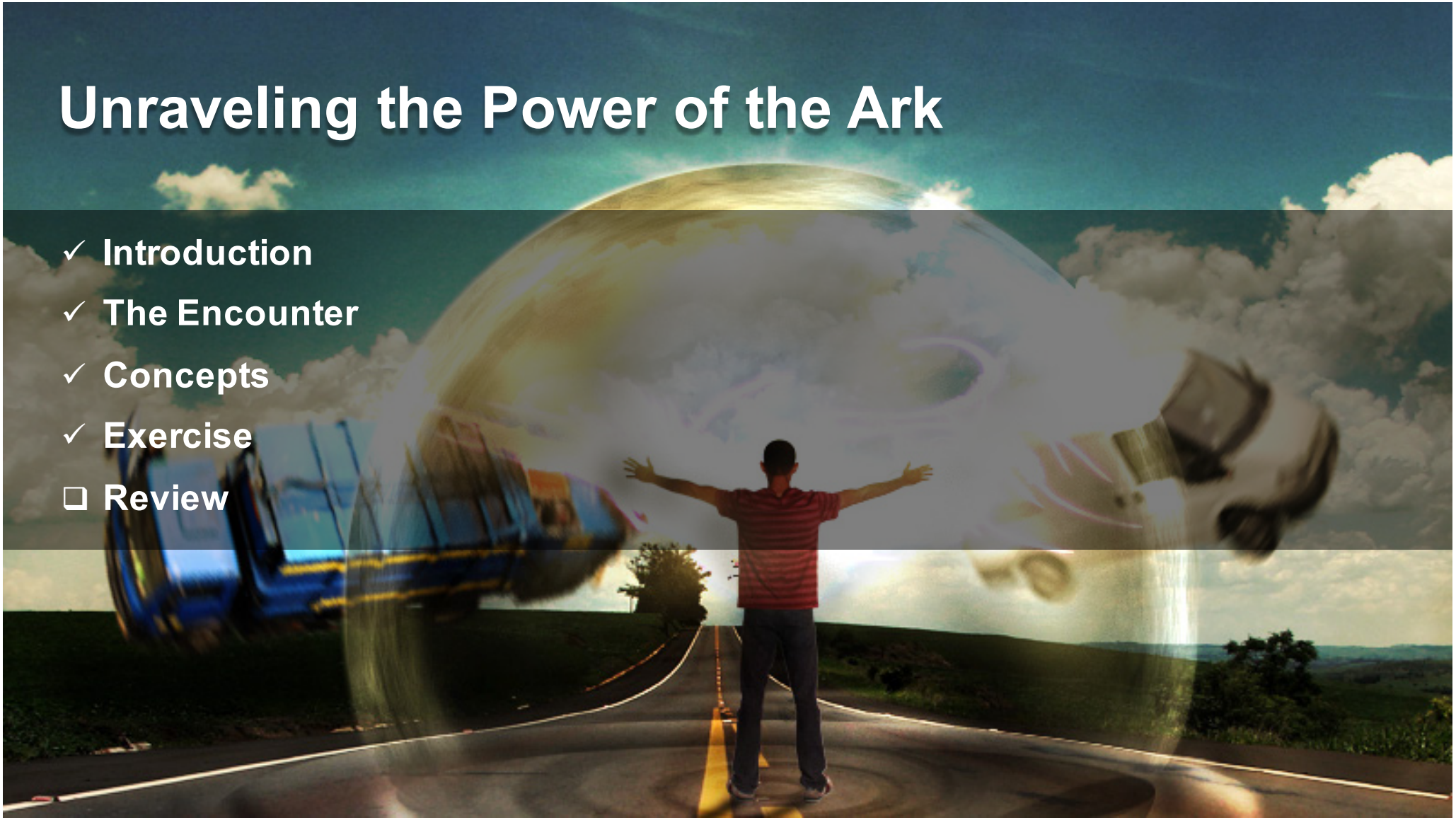
- Continue to refactor the the Ark's default recipe specification using:

- **Power Word: let**
- **Conjure shared_examples**
- **Greater helper method**
- **Ray of shared_context**
- **alias_example_group_to**

# Success

All tests pass and the size of the recipe specification has been decreased.

# Unraveling the Power of the Ark



- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ❑ Review

# Change in to the Cookbook Directory

```
> cd ~/ark
```





CHEF

# Execute the Test Suite

```
> chef exec rspec spec/recipes/default_spec.rb
```

```
..................

Finished in 3.58 seconds (files took 2.73 seconds to load)
19 examples, 0 failures



ChefSpec Coverage report generated...

  Total Resources:   41
  Touched Resources: 41
  Touch Coverage:    100.0%
```

# Edit the Recipe Specification File

~/ark/spec/recipes/default_spec.rb

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on a...d platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end

    it 'installs necessary packages' do
```

DEMO

# Live Demonstration

**https://goo.gl/ESxebk**

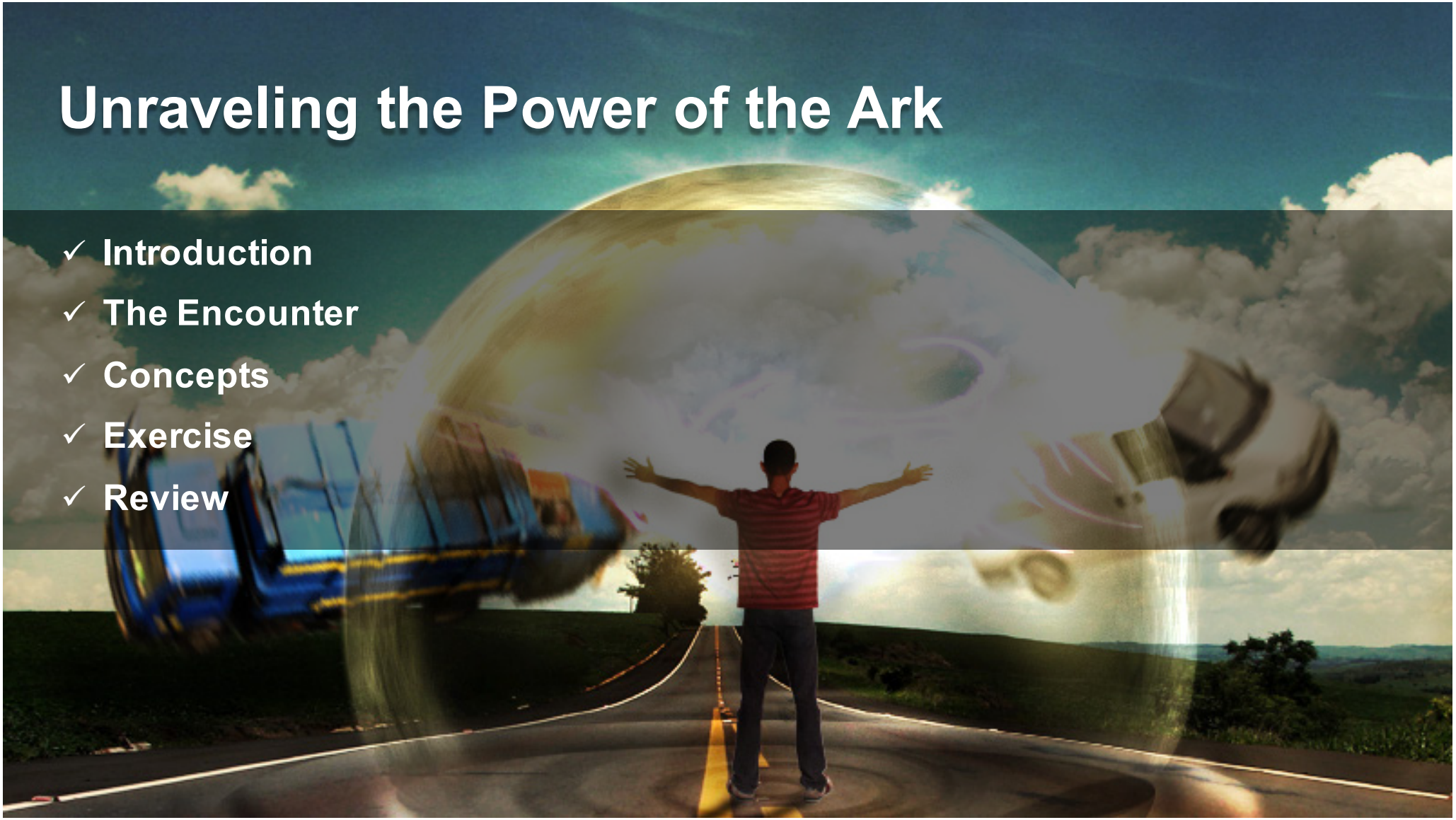
**https://goo.gl/9mRNID**





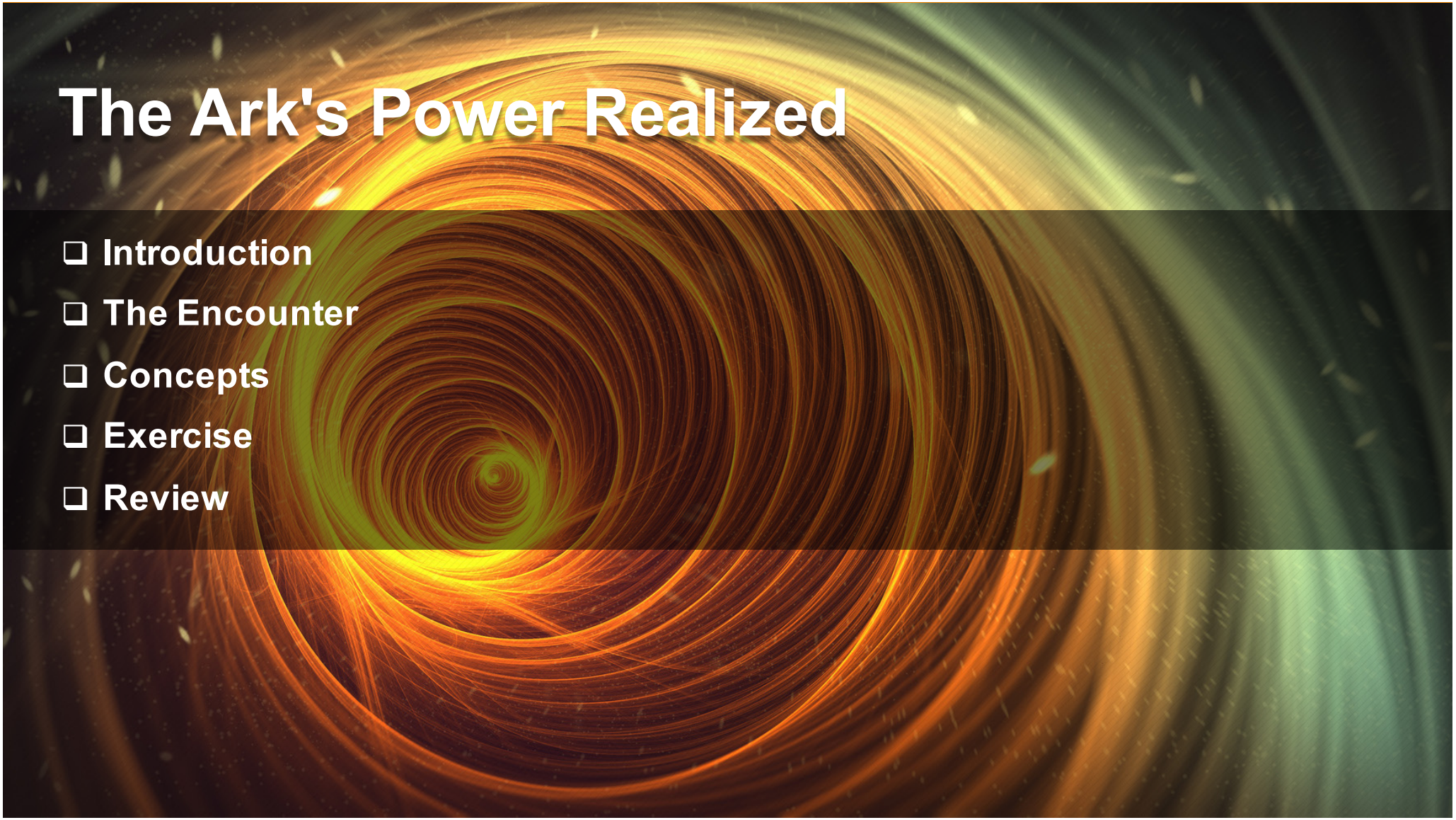
CHEF

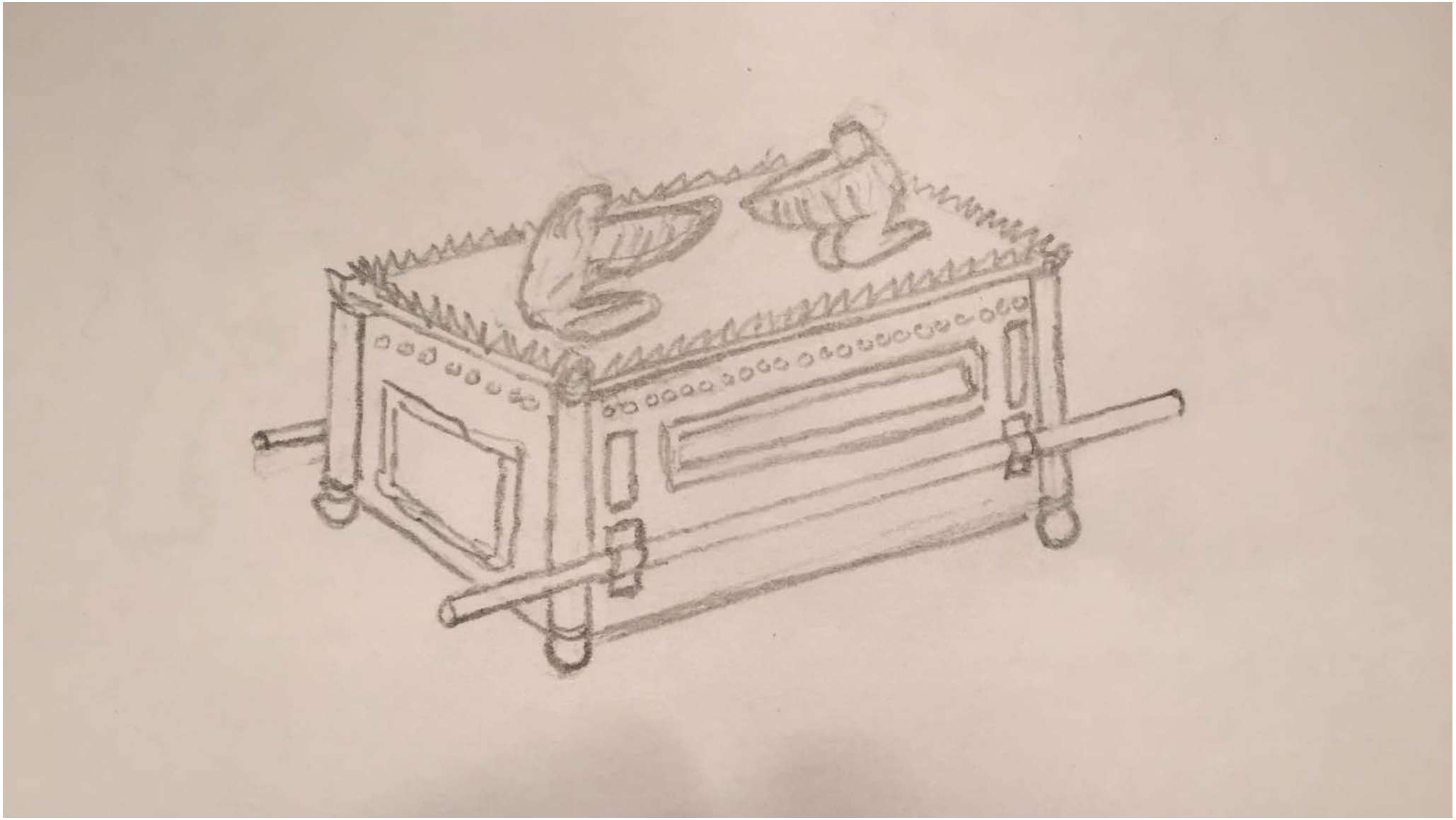
# Unraveling the Power of the Ark

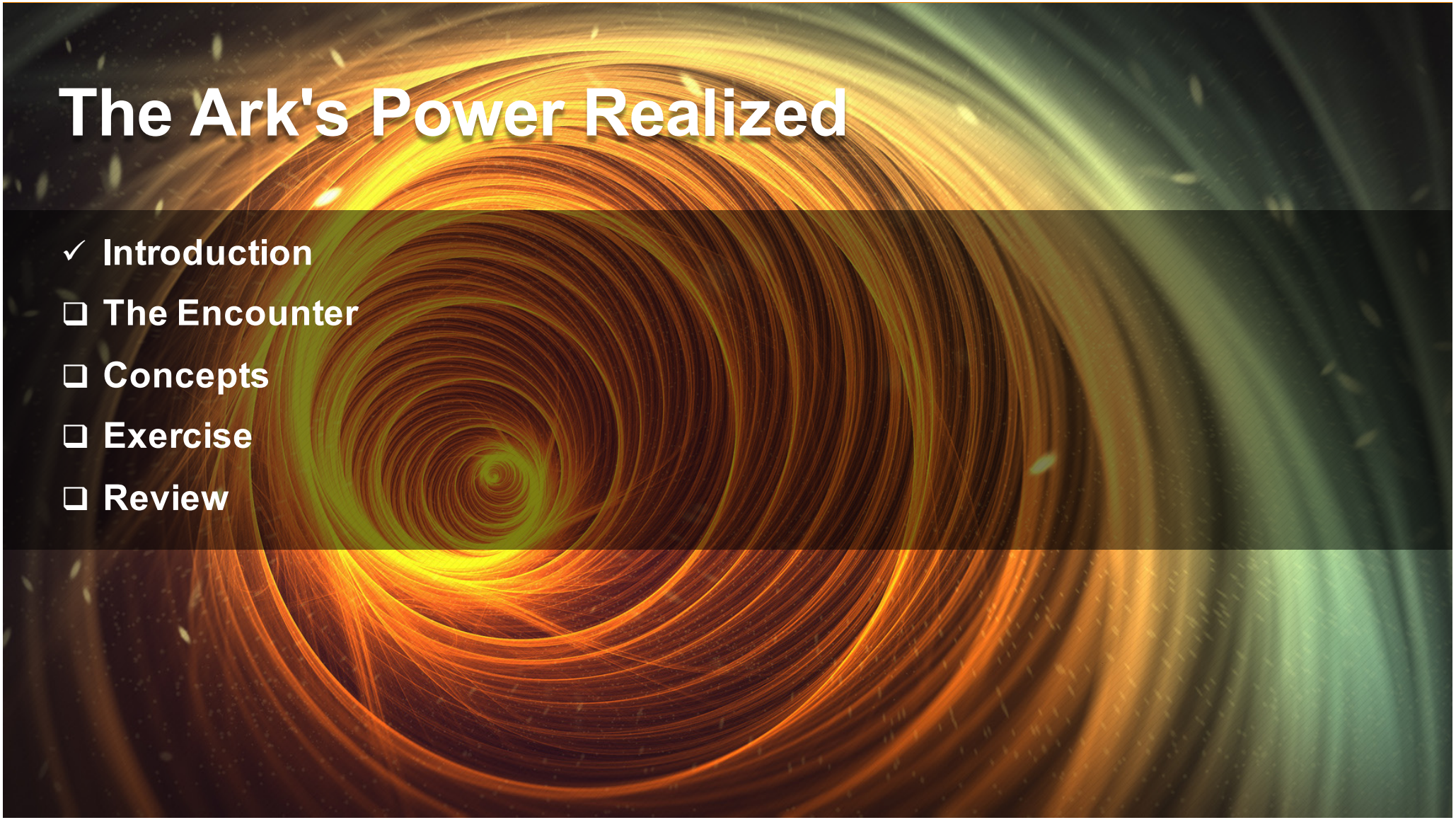


- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ✓ Review

CHEF™


The Ark's Power Realized
Introduction
The Encounter
Concepts
Exercise
Review




The Ark's Power Realized
✓ Introduction
❑ The Encounter
❑ Concepts
❑ Exercise
❑ Review

MOTIVATION

# How is the Ark Used?

The default recipe describes the components necessary for Ark to work but not how it is used.

To learn how to use the Ark one must employ it. But instead of simply unleashing it we should write a test to ensure it can do what we want it to do ... which is build an application from source.

**Adventurers your goal is to write a test that demonstrates the install_with_make action of Ark.**

# Viewing the Features of the Ark

~/ark/resources/default.rb

```
actions :install, :dump, :cherry_pick, :put, :install_with_make,
:configure, :setup_py_build, :setup_py_install, :setup_py, :unzip

default_action :install

attr_accessor :path, :release_file, :prefix_bin, :prefix_root,
:home_dir, :extension, :version

attribute :owner, :kind_of => String, :default => 'root'
attribute :group, :kind_of => [String, Fixnum], :default => 0
attribute :url, :kind_of => String, :required => true
attribute :path, :kind_of => String, :default => nil
attribute :full_path, :kind_of => String, :default => nil
```

# Understanding the Path of an Action

```
action :install_with_make do

1 directory new_resource.path

2 remote_file new_resource.release_file do
    notifies :run, "execute[unpack #{new_resource.release_file}]"
  end
```

1. Creates a directory to do work (directory)
2. Retrieves an application source (remote_file)

# Understanding the Path of an Action

```
3 execute "unpack #{new_resource.release_file}" do
    notifies :run, "execute[set owner on #{new_resource.path}]"
    notifies :run, "execute[autogen #{new_resource.path}]"
    notifies :run, "execute[configure #{new_resource.path}]"
    notifies :run, "execute[make #{new_resource.path}]"
    notifies :run, "execute[make install #{new_resource.path}]"
    action :nothing
  end
```

3 Extracts the source (execute)

# Understanding the Path of an Action

```
4 execute "set owner on #{new_resource.path}"
5 execute "autogen #{new_resource.path}"
6 execute "configure #{new_resource.path}"
7 execute "make #{new_resource.path}"
8 execute "make install #{new_resource.path}"
end
```

4. Sets the owner of the source (execute)
5. Autogen the source - if needed (execute)
6. Configures the source (execute)
7. Makes the source (execute)
8. Installs the source (execute)

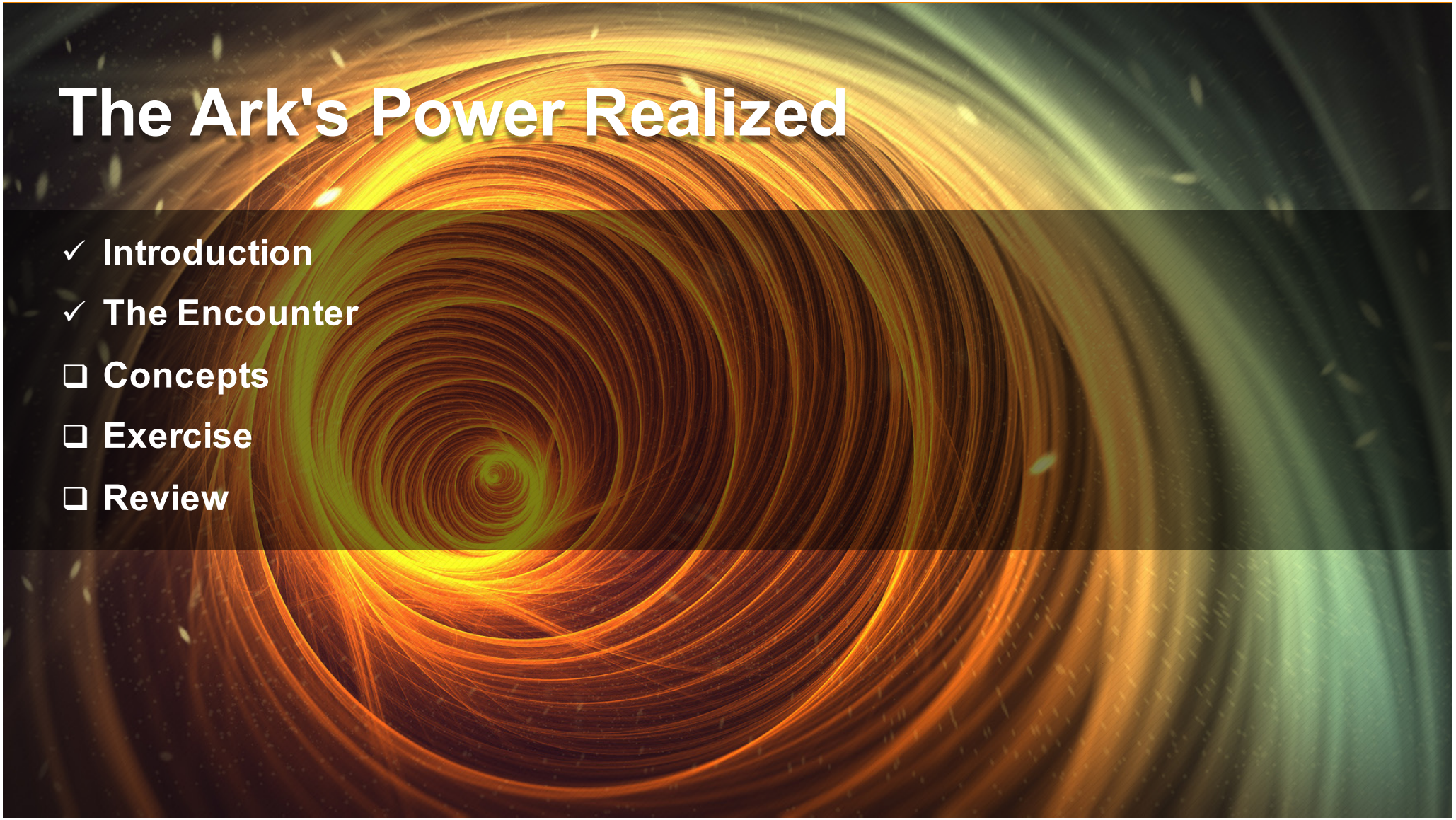
MOTIVATION

# How is the Ark Used?

The default recipe describes the components necessary for Ark to work but not how it is used.

To learn how to use the Ark one must employ it. But instead of simply unleashing it we should write a test to ensure it can do what we want it to do ... which is build an application from source.

**Adventurers your goal is to write a test that demonstrates the install_with_make action of Ark.**


The Ark's Power Realized
✓ Introduction
✓ The Encounter
❑ Concepts
❑ Exercise
❑ Review

CONCEPT

# Fixture Cookbook

Defining a cookbook, within a cookbook's test directory, that employs the cookbook in the various ways in which it may be used by downstream cookbooks.

This is often to ensure test coverage for features not in the defined recipes and their associated tests. Common if you are building a cookbook that employs a custom resource.

# Creating the Fixture Cookbook

```
> cd ~/ark
> mkdir spec/fixtures
> chef generate cookbook spec/fixtures/ark_spec
```

# Updating the Dependencies to Fixture Cookbook

~/ark/Berksfile

```
source "http://api.berkshelf.com"

metadata

cookbook 'ark_spec', path: 'spec/fixtures/ark_spec'
```

CONCEPT

# Defining Test Recipes

Within the fixture cookbook you now would define recipes that employ the custom resource is the most essential ways.





CHEF

# Viewing All the Defined Recipes

```
> ls spec/fixtures/ark_spec/recipes
```

```
spec/fixtures/ark_spec/recipes
├── cherry_pick.rb
├── cherry_pick_with_zip.rb
├── configure.rb
├── dump.rb
├── install.rb
├── install_no_stripping.rb
├── install_stripping.rb
├── install_windows.rb
├── install_with_append_env_path.rb
```

# Viewing the install Recipe

~/ark/spec/fixtures/ark_spec/recipes/install.rb

```
ark 'test_install' do
  url 'https://github.com/burtlo/ark/raw/master/fil.../foo.tar.gz'
  checksum '5996e676f17457c823d86...81e70d6a0e5f8e45b51e62e0c52e8'
  version '2'
  prefix_root '/usr/local'
  owner 'foobarbaz'
  group 'foobarbaz'
  action :install
end
```

CONCEPT

# Defining the Test for Each Recipe

For every recipe you write you now need to define a test for them.





CHEF

# Viewing the Tests for each Fixture Recipe

~/ark/spec/resources/default_spec.rb

```
require 'spec_helper'

describe 'ark' do

  # ... helpers and setup ...

  describe 'install' do
    let(:described_recipe) { 'ark_spec::install' }

    it 'generates the expected ... actions and notifications' do
      expect(chef_run).to install_ark('test_install')
```

CONCEPT

# Defining ChefSpec Matchers

ChefSpec exposes the ability for cookbook authors to package custom matchers inside a cookbook so that other developers may take advantage of them in testing.

https://goo.gl/lSQDXD

# Defining a ChefSpec Resource Matcher

~/ark/libraries/matchers.rb

```
if defined?(ChefSpec)
  def install_ark(resource_name)
    ChefSpec::Matchers::ResourceMatcher.new(:ark, :install, resource_name)
  end

  # TODO: Define additional matchers for each supported action ...
  #
  # actions :install, :install_with_make, :dump, :cherry_pick, :put
  #         :configure, :setup_py_build, :setup_py_install, :setup_py, :unzip
end
```

CONCEPT

# ChefSpec's step_into

ChefSpec overrides all providers to take no action (otherwise it would actually converge your system). This means that the steps inside your custom resource are not actually executed. If a custom resource performs actions, those actions are never executed or added to the resource collection.

In order to run the actions exposed by your custom resource, you have to explicitly tell the Runner to step into it.

https://goo.gl/SKM4L3

# Stepping to the Custom Resource

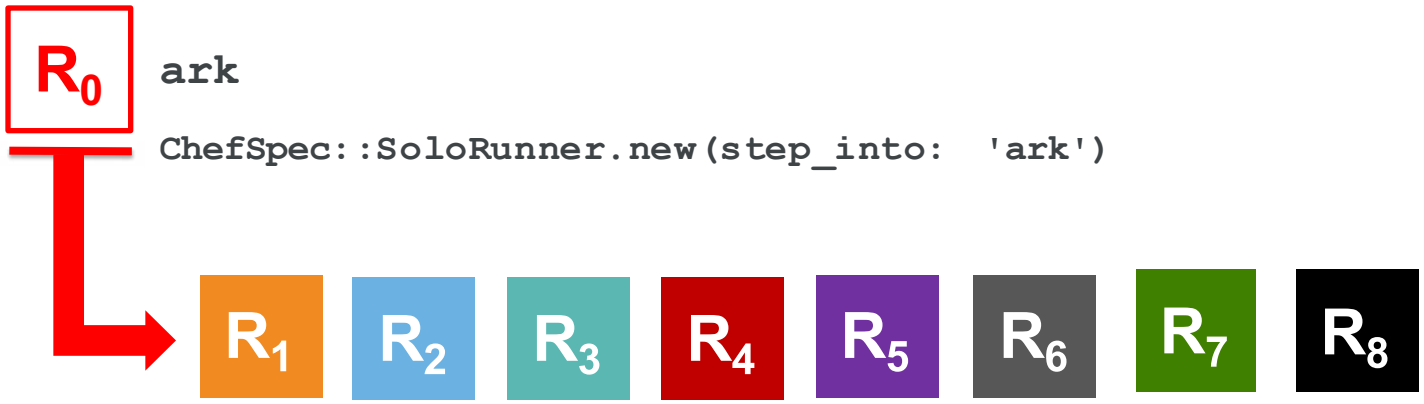
~/ark/spec/resources/default_spec.rb

```
require 'spec_helper'

describe 'ark' do
  let(:chef_run) do
    runner =
ChefSpec::SoloRunner.new(node_attributes.merge(settings).merge(step_into))
    runner.converge(described_recipe)
    runner
  end

  # This is required to test the internal components of a LWRP / Custom Resource
  # @see https://github.com/sethvargo/chefspec#testing-lwrps
  let(:step_into) do
    { step_into: ['ark'] }
  end
```

# Stepping Into Gives You Visibility Into the Resource



# Stepping into Allows Testing of Resources

~/ark/spec/resources/default_spec.rb

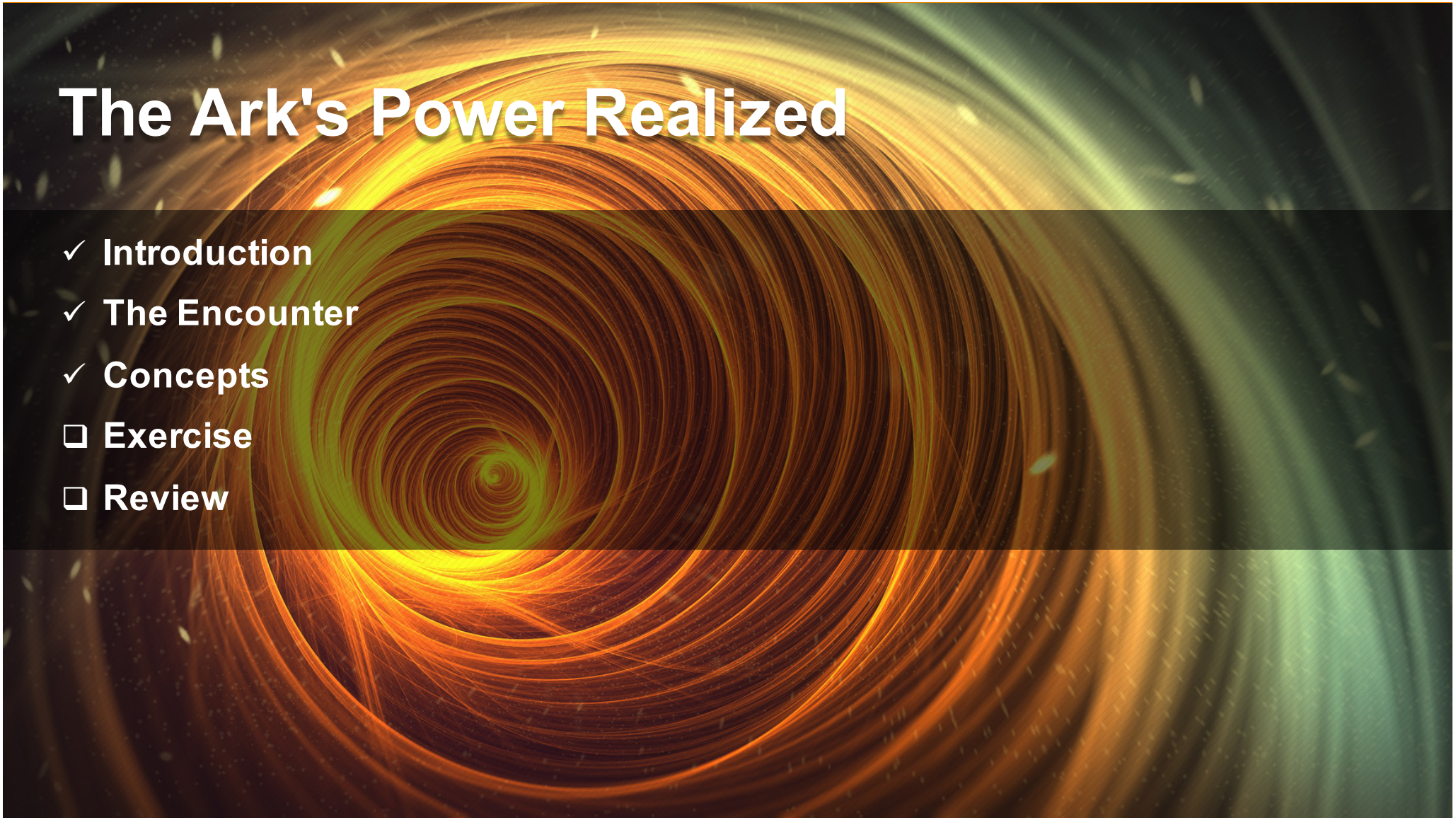
```
describe 'install' do
  let(:described_recipe) { 'ark_spec::install' }

  it 'generates the expected resources with the expected acti... notifications' do
    expect(chef_run).to install_ark('test_install')

1   expect(chef_run).to create_directory('/usr/local/test_install-2')
    resource = chef_run.directory('/usr/local/test_install-2')
    expect(resource).to notify('execute[unpack /var/chef/cache/test_install-2.t...

2   expect(chef_run).to create_remote_file('/var/chef/cache/test_install-2.tar....
    resource = chef_run.remote_file('/var/chef/cache/test_install-2.tar.gz')
    expect(resource).to notify('execute[unpack /var/chef/cache/test_install-2.t...

    # ... testing other resources within the custom resource action ...
```


The Ark's Power Realized
✓ Introduction
✓ The Encounter
✓ Concepts
❑ Exercise
❑ Review

ENCOUNTER

# Exercise

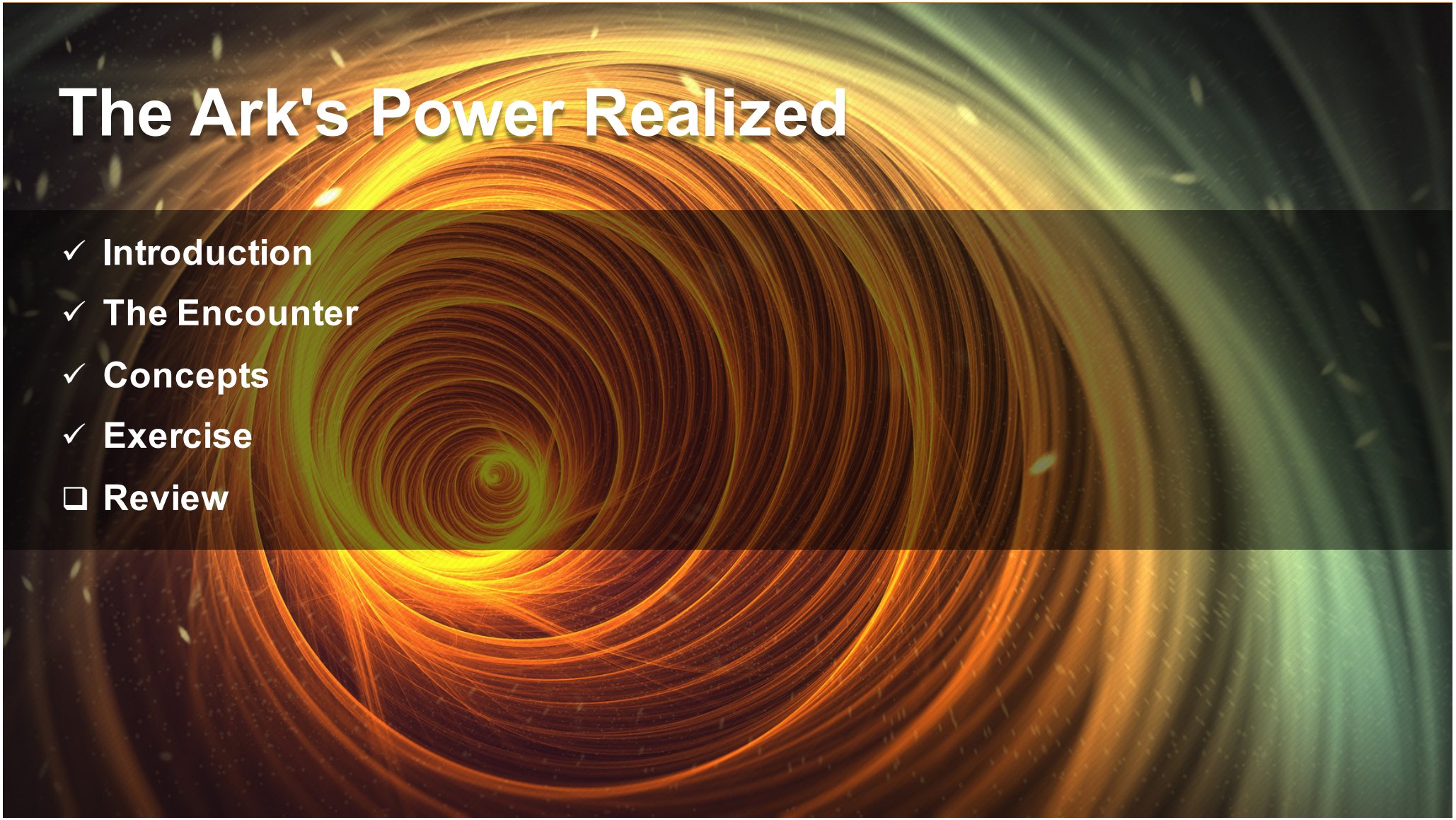
- Define an `install_with_make` ChefSpec Resource Matcher
- Edit `~/ark/spec/resources/default_spec.rb` and define the 'install_with_make' example
- Run `chef exec rspec spec/resources/default_spec.rb`

# Success

The 'install_with_make' example passes with 100% coverage



CHEF


The Ark's Power Realized
✓ Introduction
✓ The Encounter
✓ Concepts
✓ Exercise
❑ Review

# Change in to the Cookbook Directory

```
> cd ~/ark
```





CHEF

# Execute the Test Suite

```
> chef exec rspec spec/resources/default_spec.rb:57
```

# Edit the Resource Specification File

~/ark/spec/resources/default_spec.rb

```
require 'spec_helper'

describe 'ark' do
  let(:chef_run) do
    runner = ChefSpec::SoloRunner.new(node_attributes.merge(set...
    runner.converge(described_recipe)
    runner
  end

  let(:node_attributes) do
    {}
  end
```

DEMO

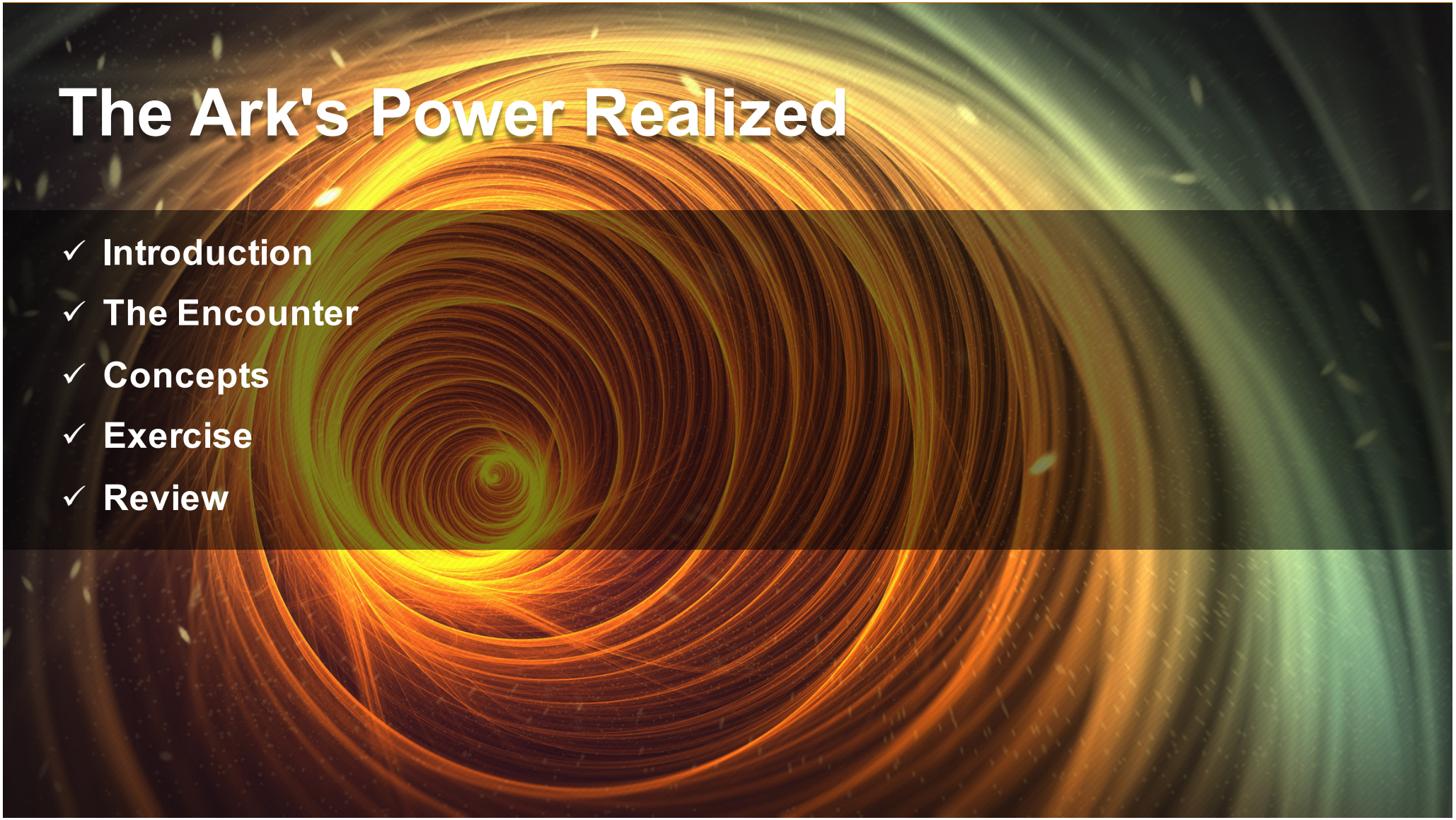
# Live Demonstration

**https://goo.gl/DIjNcZ**





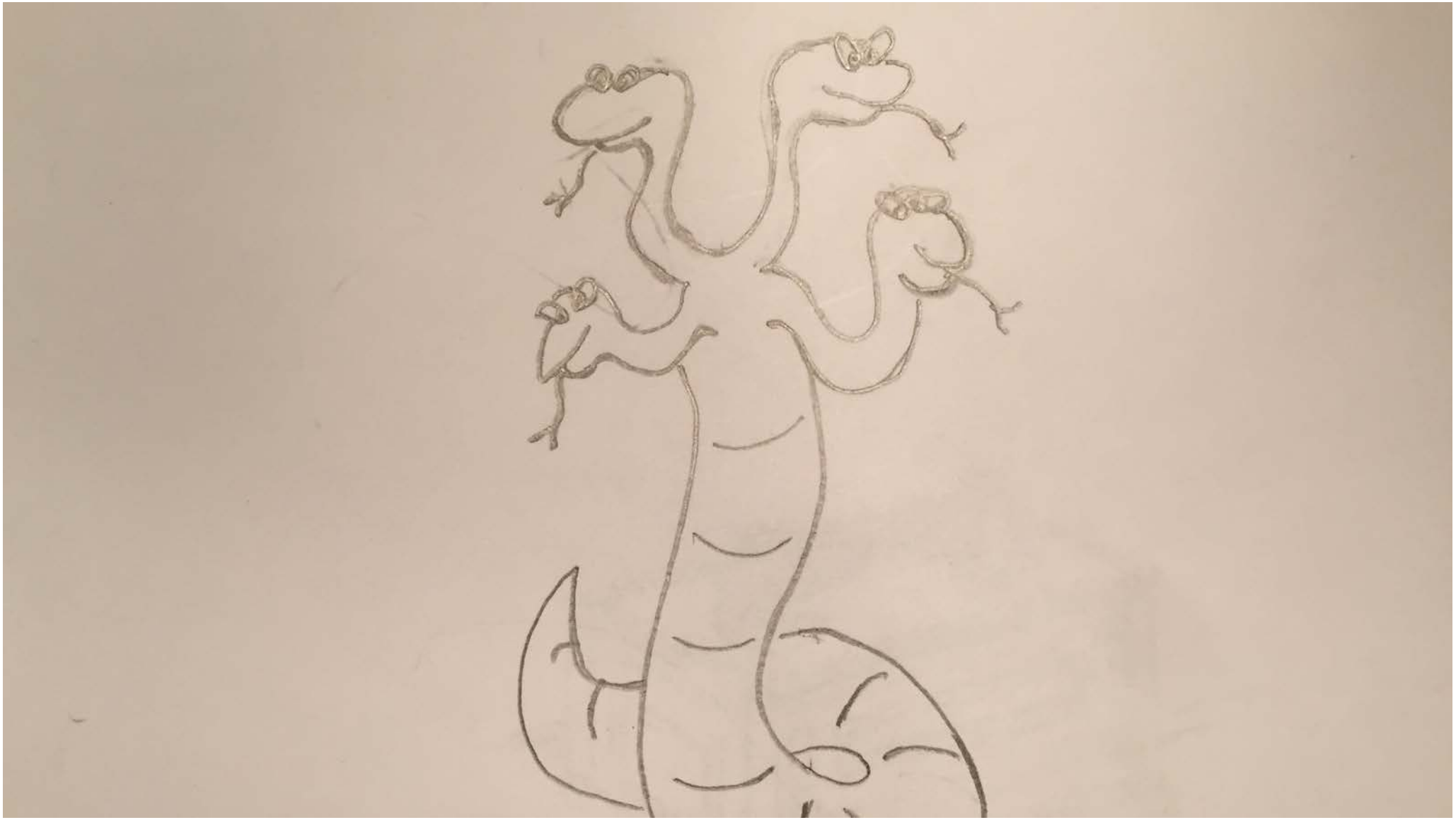
CHEF


The Ark's Power Realized
✓ Introduction
✓ The Encounter
✓ Concepts
✓ Exercise
✓ Review

CHEF™

# The Python's Scales

- ❑ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review




The Python's Scales
✓ Introduction
❑ The Encounter
❑ Concepts
❑ Exercise
❑ Review

# Viewing the Python's Default Recipe

~/python/recipes/default.rb

```
remote_file '/tmp/Python-3.4.4.tgz' do
  source 'https://www.python.org/ftp/python/3.4.4/Python-3.4.4.tgz'
  notifies :run, 'execute[extract python]', :immediately
end

execute 'extract python' do
  command 'tar xzf /tmp/Python-3.4.4.tgz'
  cwd '/tmp'
  action :nothing
  notifies :run, 'execute[python build and install]', :immediately
end
# ... CONTINUES ON THE NEXT SLIDE ...
```

# Viewing the Python's Default Recipe

~/python/recipes/default.rb

```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...
execute 'extract python' do
  command 'tar xzf /tmp/Python-3.4.4.tgz'
  cwd '/tmp'
  action :nothing
  notifies :run, 'execute[python build and install]', :immediately
end

execute 'python build and install' do
  command './configure && make install'
  cwd '/tmp/Python-3.4.4'
  action :nothing
end
```

## Python's default Recipe

- **Retrieve the Python source (remote_file)**
- **Extracts the source (execute)**
- **Configures the source (execute)**
- **Installs the source (execute)**

## Ark's install_with_make

- Creates a directory to do work (directory)
- **Retrieves an application source (remote_file)**
- **Extracts the source (execute)**
- Sets the owner of the source (execute)
- Autogen the source - if needed (execute)
- **Configures the source (execute)**
- Makes the source (execute)
- **Installs the source (execute)**

MOTIVATION

# Using the Ark to Slay the Python

Replacing these three resources with a single resource will make the recipe much more readable and possibly more maintainable.

**Adventurers your FIRST goal is to replace the existing installation method with the Ark resource.**





CHEF

MOTIVATION

# Using the Python's Pip as a Weapon

When Python is installed it also installs a tool called **pip** that allows you to install packages. Cookbooks that rely on the python cookbook will likely need Python installed and need access to this tool.

**Adventurers your Second goal is to create a custom resource that allows you to leverage the pip tool in other cookbooks.**





CHEF


The Python's Scales
✓ Introduction
✓ The Encounter
❑ Concepts
❑ Exercise
❑ Review

CONCEPT

# Custom Resource

A custom resource is a simple extension of Chef that is implemented as part of a cookbook. It follows easy, repeatable syntax patterns and effectively leverages resources that are built into Chef.

The result is reusable in the same way as resources that are built into Chef.

https://docs.chef.io/custom_resources.html





CHEF

~/python/spec/fixtures/python_spec/recipes/pip_install.rb

```
pip 'django'
```

~/python/spec/fixtures/python_spec/recipes/pip_install.rb

```
pip 'django' do
  action :install
end
```

~/python/spec/fixtures/python_spec/recipes/pip_install.rb

```
pip 'django' do
  package_name 'django'
  action :install
end
```

# Changing into the Directory of the Cookbook

```
> cd ~/python
```

# Generating an LWRP in the Cookbook

```
> chef generate lwrp pip
```





CHEF

# Removing the Unnecessary Providers Directory

```
> rm -rf providers
```

# Defining the name of the resource

~/python/resources/pip.rb

```
resource_name :pip
```

CONCEPT

# Custom Resource: action

Each custom resource must have at least one action that is defined within an action block.

The first action defined in the resource definition is considered the default action.

https://docs.chef.io/custom_resources.html#define-actions

# Defining an Action

~/python/resources/pip.rb

```
resource_name :pip

action :install do
  # ... contains the resources needed to converge
end
```

# Defining an Action

~/python/resources/pip.rb

```
resource_name :pip

action :install do
  execute '/usr/local/bin/pip3 install PACKAGENAME'
end
```

CONCEPT

# Custom Resource: property

Custom properties are defined in the resource. A property describes a field that a user of the custom resource can set when they define the custom resource within their recipes.

These properties have a type, can have default values, or be set to retrieve the value from the name of the resource.

https://docs.chef.io/custom_resources.html#define-properties

# Defining an Action

~/python/resources/pip.rb

```
resource_name :pip

property :package_name, String

action :install do
  execute "/usr/local/bin/pip3 install #{package_name}"
end
```

CONCEPT

# Custom Resource: name_property

Custom properties are defined in the resource. A property describes a field that a user of the custom resource can set when they define the custom resource within their recipes.

These properties have a type, can have default values, **or be set to retrieve the value from the name of the resource**.

https://docs.chef.io/custom_resources.html#define-properties

# Defining an Action

~/python/resources/pip.rb

```
resource_name :pip

property :package_name, String, name_property: true

action :install do
  execute "/usr/local/bin/pip3 install #{package_name}"
end
```


The Python's Scales
✓ Introduction
✓ The Encounter
✓ Concepts
❑ Exercise
❑ Review

ENCOUNTER

# Exercise

- Refactor the Python cookbook's default recipe to use the ark resource to install_with_make Python 3.4.4
- Define a custom resource, named `pip`

# Success

All tests pass for the Python cookbook with 100% coverage



CHEF


The Python's Scales
✓ Introduction
✓ The Encounter
✓ Concepts
✓ Exercise
❑ Review

DEMO

# Live Demonstration

**https://goo.gl/Etc9Gv**

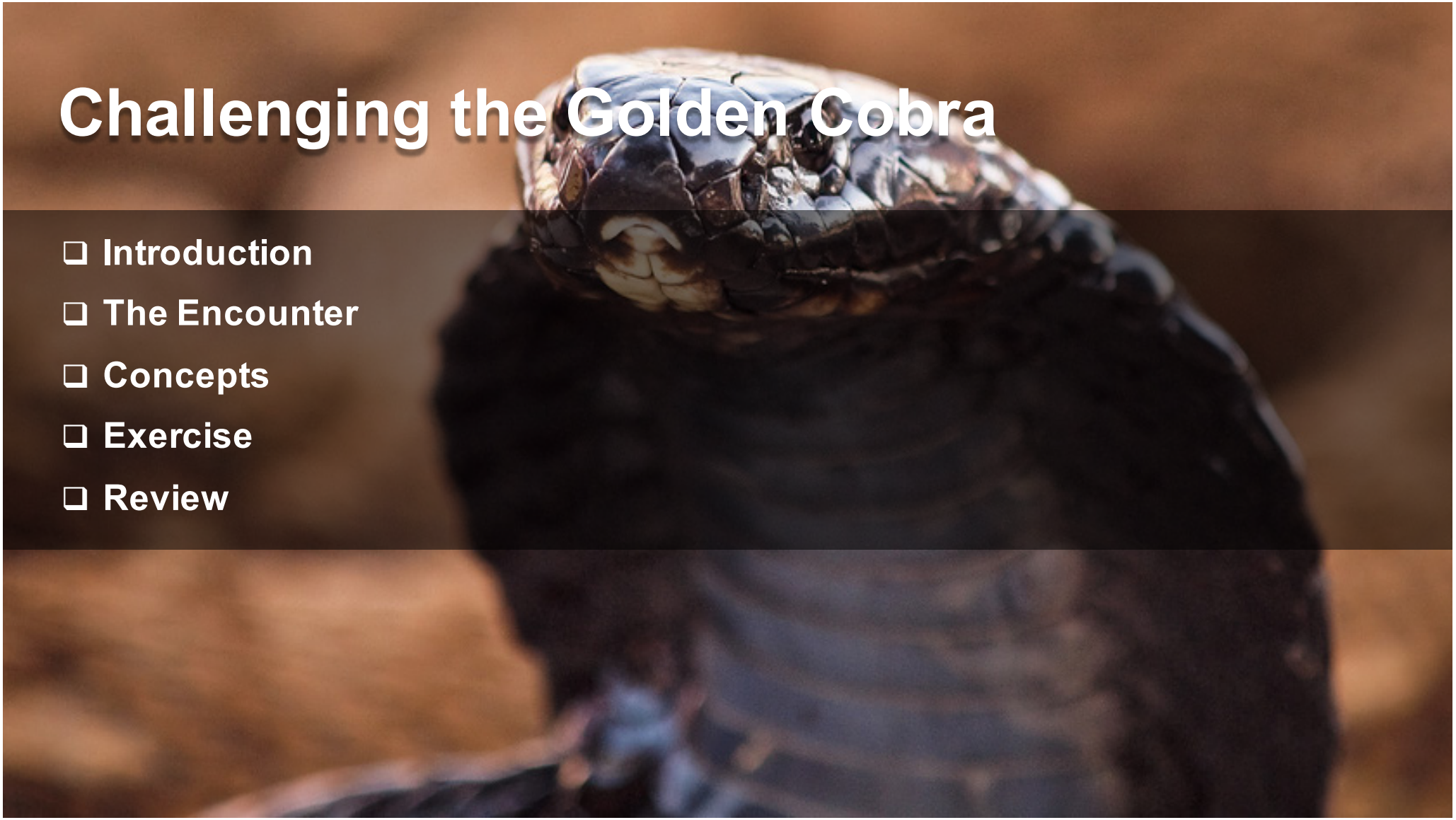




CHEF

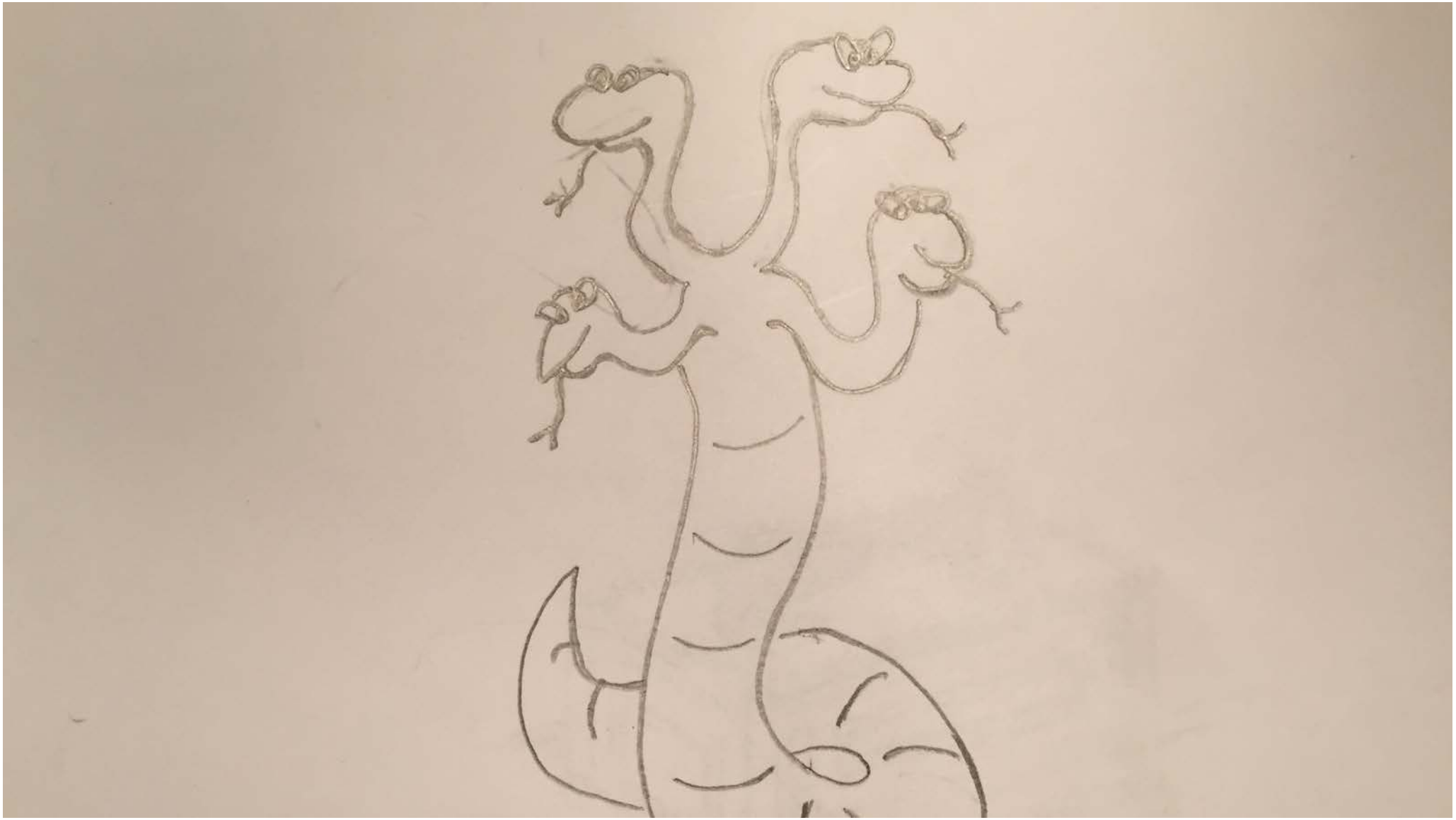

The Python's Scales
✓ Introduction
✓ The Encounter
✓ Concepts
✓ Exercise
✓ Review

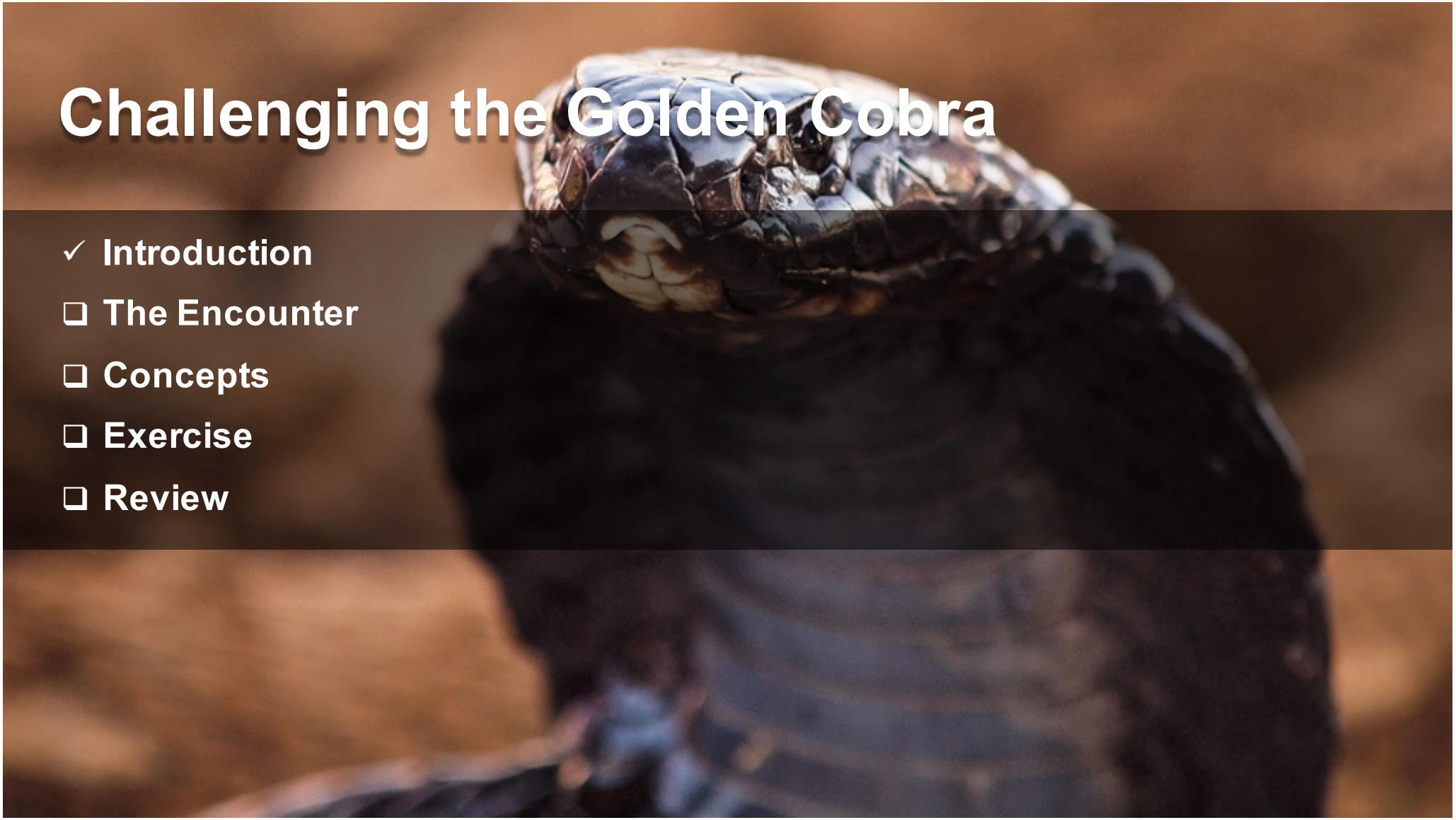
CHEF™



# Challenging the Golden Cobra

- ❑ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review





# Challenging the Golden Cobra

- ✓ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

MOTIVATION

# The Golden Cobra Has Requirements

The golden_cobra cookbook is an application cookbook that deploys a Django application. It needs Python installed, a number of Python packages, and relies on a data bag.

**Adventurers your goal is to finish the golden_cobra!**





CHEF

# Viewing the Existing Default Recipe

~/golden_cobra/recipes/default.rb

```
execute 'yum update -y'

package 'sqlite-devel'

# TODO: install python 3.4.4
# TODO: install python packages: django, uwsgi, and gunicorn

directory '/sites'

package 'git'
# ... CONTINUES ON THE NEXT SLIDE ...
```

# Viewing the Existing Default Recipe

~/golden_cobra/recipes/default.rb

```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...

search('site','*:*').each do |site_data|

  git "/sites/#{site_name}" # ... collapsed definitions
  execute '/usr/local/bin/python3 manage.py migrate' #...
  execute "gunicorn #{site_name}.wsgi -D -b #{site_bind}" #...

end
```

MOTIVATION

# The Golden Cobra Has Requirements

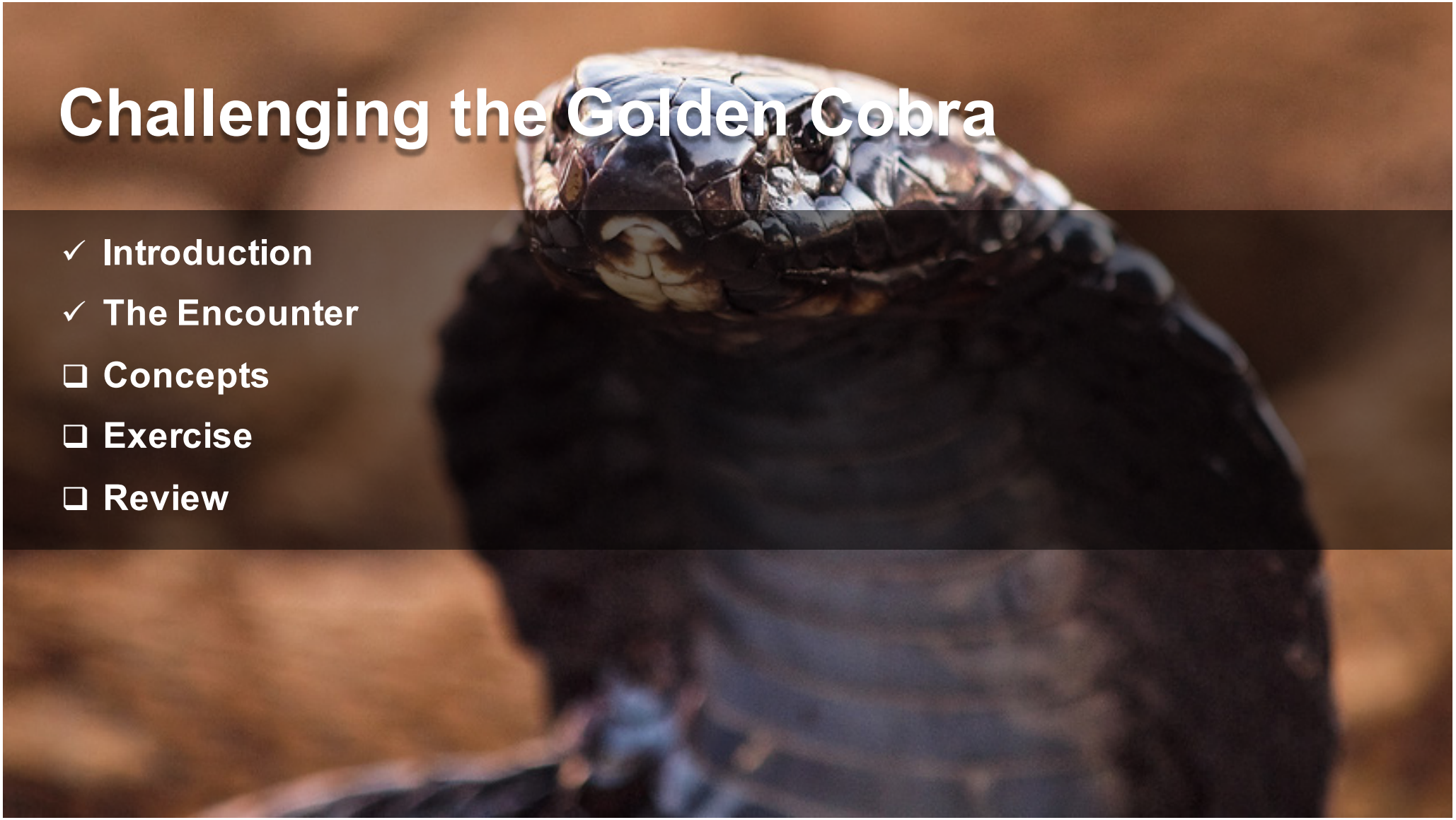
The golden_cobra cookbook is an application cookbook that deploys a Django application. It needs Python installed, a number of Python packages, and relies on a data bag.

**Adventurers your goal is to finish the golden_cobra!**





CHEF



# Challenging the Golden Cobra

- ✓ Introduction
- ✓ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

CONCEPT

# Stubbing Search

Because ChefSpec is a unit-testing framework, it is recommended that all third-party API calls be mocked or stubbed. ChefSpec exposes a helpful RSpec macro for stubbing search results in your tests.

If you converge a Chef recipe that implements a search call ChefSpec will throw an error. You may stub the search call to prevent the error and test a variety of search results that could be returned.

https://github.com/sethvargo/chefspec#search

CONCEPT

# ServerRunner vs SoloRunner

There are two runners available when defining your specifications. Choosing the right one is often dependent on the context of how this cookbook/recipe will be used within your organization.

Stubbing Search and other values is different based on the runner that you choose.

# Viewing a Standard ServerRunner Definition

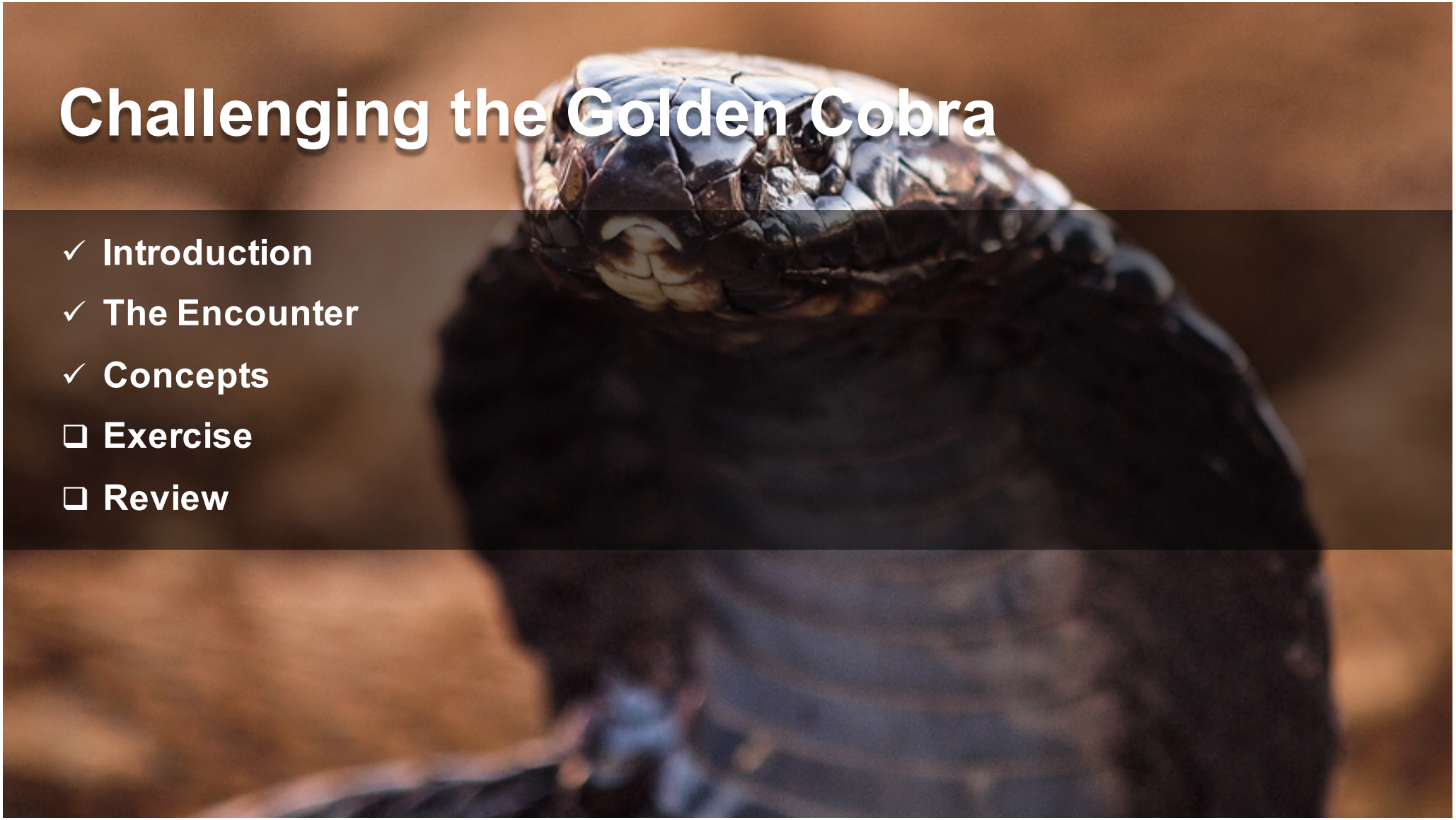
```
let(:chef_run) do
  runner = ChefSpec::ServerRunner.new
  runner.converge(described_recipe)
end
```

# Expanding the ServerRunner Definition

```
let(:chef_run) do
  runner = ChefSpec::ServerRunner.new do |node,server|
    # setup additional details about the state of the node
    # setup additional details about the state of the server
  end
  runner.converge(described_recipe)
end
```

# Creating a Data Bag on the Server

```
let(:chef_run) do
  runner = ChefSpec::ServerRunner.new do |node,server|
    server.create_data_bag('users', {
      'fwebber' => {
        'name' => 'fwebber',
        'home' => '/home/fwebber'
      }
    })
  end
  runner.converge(described_recipe)
end
```



# Challenging the Golden Cobra

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ❑ Exercise
- ❑ Review

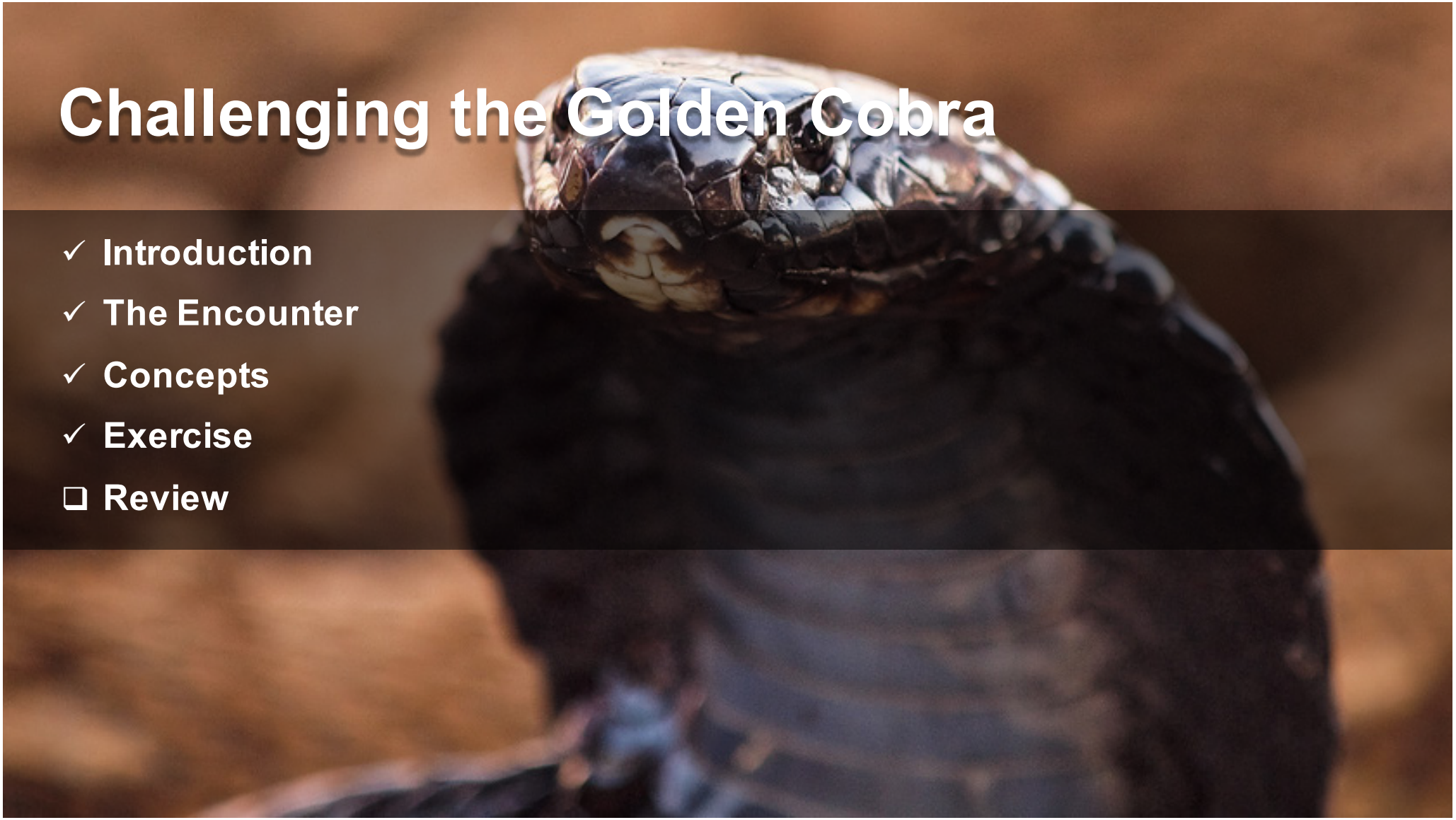
ENCOUNTER

## Exercise

- Execute the golden_cobra cookbook's tests and identify the failures
- Update the test suite to stub the search results
- Update the recipe to include needed recipes and resources

## Success

All tests pass with 100% coverage



# Challenging the Golden Cobra

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ❑ Review

DEMO

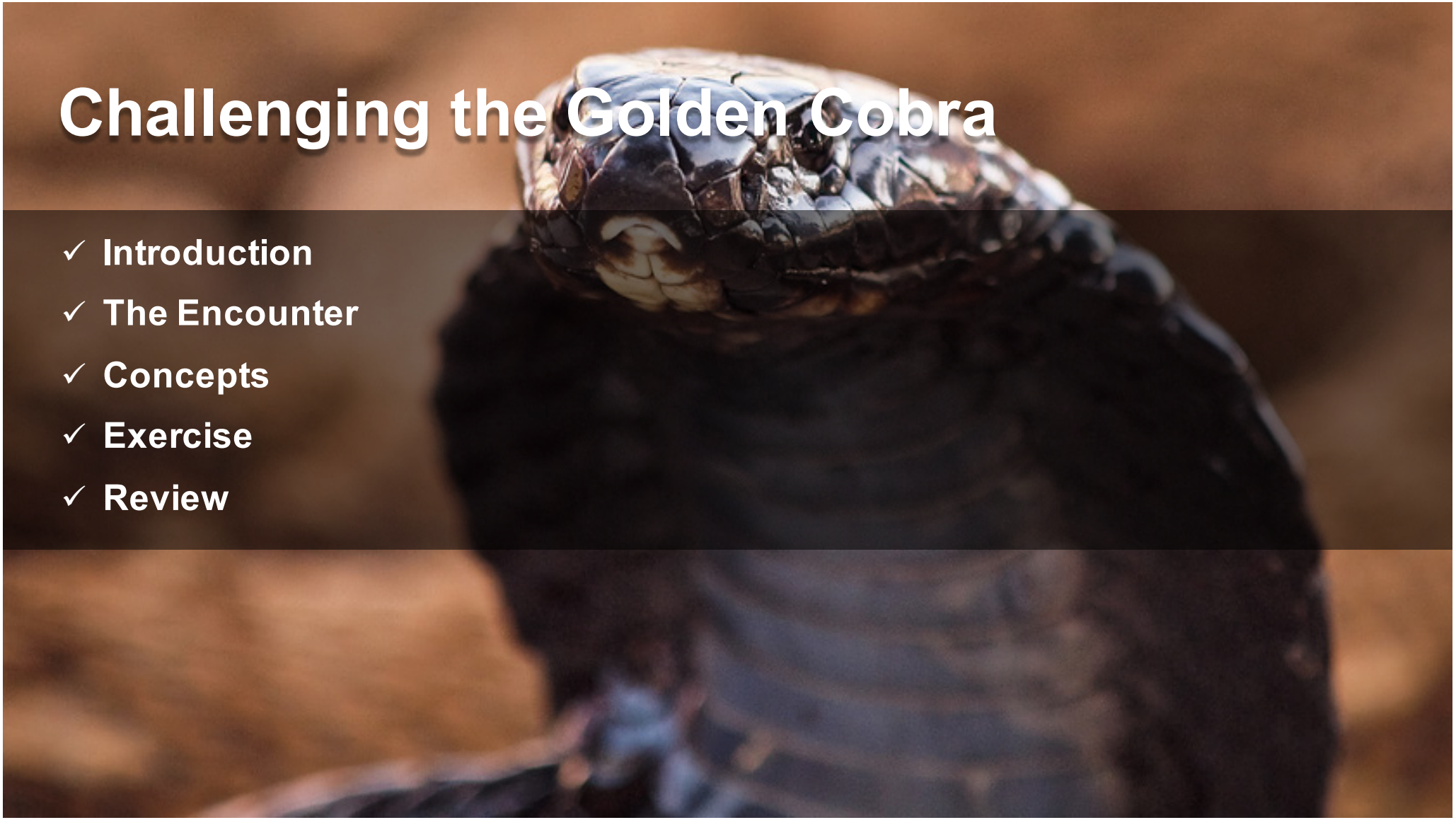
# Live Demonstration

**https://goo.gl/2vMP9k**





CHEF



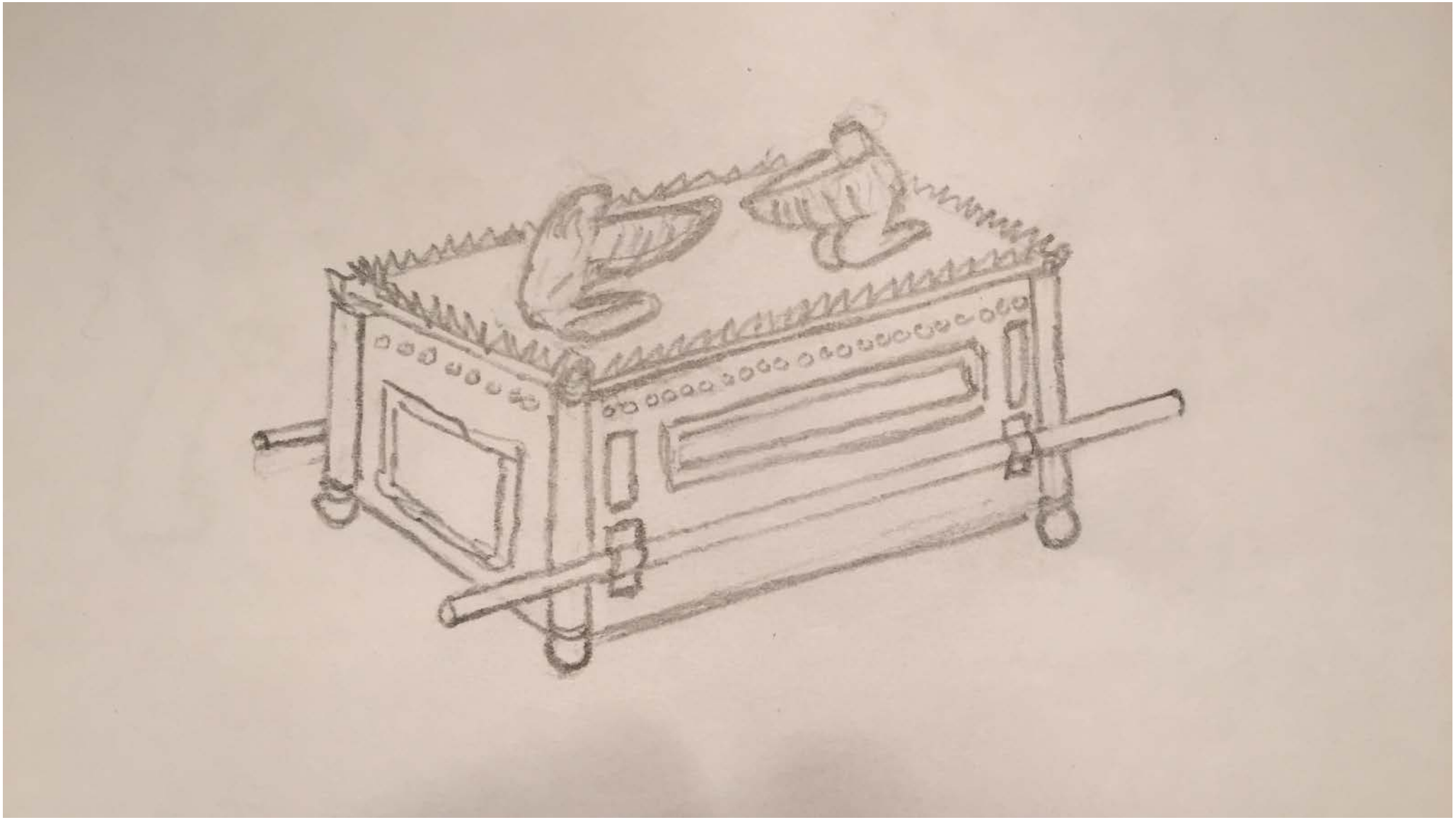
# Challenging the Golden Cobra

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ✓ Review

CHEF™

# Securing the Power of the Ark

- Introduction
- The Encounter
- Concepts
- Exercise
- Review



# Securing the Power of the Ark

- ✓ Introduction
- ❑ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

MOTIVATION

# Testing Everything Through Recipes

The recipes in the fixture cookbook do a good job testing the custom resource from end-to-end but to to test all of the code paths one would need to generate a lot more recipes. This is because the Ark cookbook relies on some complex helper methods

Instead of testing each path through the code at the recipe level we can isolate the helper and test it with a large set of inputs. Testing them in isolation will allow better coverage at the boundaries and delivers results faster.

**Adventurers your goal is to test a single helper method!**





CHEF

# Securing the Power of the Ark

- ✓ Introduction
- ✓ The Encounter
- ❑ Concepts
- ❑ Exercise
- ❑ Review

CONCEPT

# Helper Methods

To create more concise recipes code is often defined in methods that are stored in modules. These modules are then included within the recipe or the resources when the cookbook is loaded.

These helper methods provide an easy way to simplify, on the surface, complicated logic that may need to be calculated or retrieved from the system.

CONCEPT

# Include in a Class

Methods defined within a module is portable code that can be inserted into any context that includes them. This is often done at the start of the recipe which includes the methods into the Chef::Recipe class.

This same tactic can be used within any class that you may choose to define.

# A module Defines Methods

```
module Challenges
  def monsters ; end
  def traps ; end
  def mystery ; end
  def treasure ; end
end
```

# Including a Module Grants the Object those Methods

```
module Challenges
  def monsters ; end
  def traps ; end
  def mystery ; end
  def treasure ; end
end

class Dungeon
  include Challenges
end

dungeon = Dungeon.new
dungeon.monsters
```

# Including a Module Grants the Object those Methods

```
module Challenges
  def monsters ; end
  def traps ; end
  def mystery ; end
  def treasure ; end
end

class Castle
  include Challenges
end

castle = Castle.new
castle.monsters
```

# Viewing One Helper Method

```
module Opscode
  module Ark
    module ProviderHelpers
       def cherry_pick_tar_command(tar_args)
        cmd = node['ark']['tar']
        cmd += " #{tar_args}"
        cmd += " #{new_resource.release_file}"
        cmd += " -C"
        cmd += " #{new_resource.path}"
        cmd += " #{new_resource.creates}"
        cmd += tar_strip_args
        cmd
      end
    end
  end
end
```

# Including the Module in a Class

```
module Opscode
  module Ark
    module ProviderHelpers
      def cherry_pick_tar_command(tar_args)
        # ... implementation
      end
   end
  end
end

class Subject
  include Opscode::Ark::ProviderHelpers
end
```

# The Subject Class Does Not Have Everything

```
class Subject
  include Opscode::Ark::ProviderHelpers
end

subject.cherry_pick_tar_command(tar_args)

# ERROR subject does not have the method node
# ERROR subject does not have the method tar_strip_args
# ERROR subject does not have the method new_resource
```

# Helper Method has Methods it Calls

```
module Opscode
  module Ark
    module ProviderHelpers
       def cherry_pick_tar_command(tar_args)
        cmd = node['ark']['tar']
        cmd += " #{tar_args}"
        cmd += " #{new_resource.release_file}"
        cmd += " -C"
        cmd += " #{new_resource.path}"
        cmd += " #{new_resource.creates}"
        cmd += tar_strip_args
        cmd
      end
    end
  end
end
```

# Adding More Methods to the Subject

```
class Subject
  include Opscode::Ark::ProviderHelpers
  def node
    { 'ark' => { 'tar' => '/bin/tar' } }
  end

  def tar_strip_args
    'sure_why_not_strip_the_args'
  end

  def new_resource
    # This is where it gets more complicated ...
  end
end

subject.cherry_pick_tar_command(tar_args)
```

# The new_resource has it's Own Methods

```
module Opscode
  module Ark
    module ProviderHelpers
       def cherry_pick_tar_command(tar_args)
        cmd = node['ark']['tar']
        cmd += " #{tar_args}"
        cmd += " #{new_resource.release_file}"
        cmd += " -C"
        cmd += " #{new_resource.path}"
        cmd += " #{new_resource.creates}"
        cmd += tar_strip_args
        cmd
      end
    end
  end
end
```

# Creating a Fake new_resource

```
class Subject
  include Opscode::Ark::ProviderHelpers
  def node
    { 'ark' => { 'tar' => '/bin/tar' } }
  end
  def tar_strip_args
    'sure_why_not_strip_the_args'
  end
  def new_resource
    require 'ostruct'
    OpenStruct.new(release_file: 'file',
                   path: 'path',
                   creates: 'sure_why_not')
  end
end

subject.cherry_pick_tar_command(tar_args)
```

CONCEPT

# Stubs and Doubles

Generating a class for every scenario can make the code less clear to understand as it requires more boiler plate code to accomplish the goal for **every** specification.

RSpec provides a way to stub method calls which will allow you to more quickly setup the scenario that isolates the test and helps you express your expectation quicker.

No Reference





CHEF

# Defining Stub methods on an Object

```
describe Opscode::Ark::ProviderHelpers do
  class Subject
    include Opscode::Ark::ProviderHelpers
  end

  describe '#cherry_pick_tar_command' do
    let(:subject) do
      obj = Subject.new
      allow(obj).to receive(:node).and_return({ 'ark' =>
                                                {'tar' => '/bin/tar'} })
      allow(obj).to receive(:new_resource).and_return(new_resource)
      allow(obj).to receive(:tar_strip_args).and_return('')
      obj
    end
```





CHEF

# Returning a new_resource Object

```
describe Opscode::Ark::ProviderHelpers do
  class Subject
    include Opscode::Ark::ProviderHelpers
  end

  describe '#cherry_pick_tar_command' do
    let(:subject) do
      obj = Subject.new
      allow(obj).to receive(:node).and_return({ 'ark' =>
                                                {'tar' => '/bin/tar'} })
      allow(obj).to receive(:new_resource).and_return(new_resource)
      allow(obj).to receive(:tar_strip_args).and_return('')
      obj
    end
```

# Defining a Test Double for new_resource

```
describe '#cherry_pick_tar_command' do
  let(:subject) do
    # ...
  end

  let(:new_resource) do
    obj = double(:new_resource)
    allow(obj).to receive(:release_file).and_return('release-file')
    allow(obj).to receive(:path).and_return('path')
    allow(obj).to receive(:creates).and_return('creates')
    obj
  end
```

# Securing the Power of the Ark

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ❑ Exercise
- ❑ Review

ENCOUNTER

# Exercise

- Define a test for `tar_strip_args` with strip_components set to 0
- Define a test for `tar_strip_args` with strip_components set to non-zero.

# Success

All the tests that have been defined pass.

# Securing the Power of the Ark

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ❑ Review

DEMO

# Live Demonstration

**https://goo.gl/ayi0Sk**





CHEF

# Securing the Power of the Ark

- ✓ Introduction
- ✓ The Encounter
- ✓ Concepts
- ✓ Exercise
- ✓ Review

CHEF™