```
CHEF

| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is oc.is) | if running_service.com
| As(opscode-projet opscode, prof), by(oc.is) | if running_service.co
```

raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)" end res end

_by_key.default

"@resources_by_key"]

 $arg = /^.+\[(.+)\]$ resource_list = \$1 resource_list.split(",").each do |instance_name|

key = key.to_s res = @resources_by_key[key] unless res raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)"

resources = rl.merge(rl) { |k, r| Chef::Resource.from_hash(r) } collection.instance_variable_set(:@resources_by_key, resources) collection end private

instance_names.each do |instance_name| results << lookup(create_key(resource_type, instance_name)) end results end def find_resource_by_string(arg)

[lookup(create_key(resource_type, name))] end raise ArgumentError, "Bad string format #{arg}, you must have a string like resource_type[name]!" end endend#require "chef/exceptions"class Chef

fore_notification_collection.delete(resource.declared_key) run_context.immediate_notification_collection.delete(resource.declared_key) run_context.delayed_notification_collection.delete(resource.declared_key)

"The string `#{query_object}' is not valid for resource collection lookup. Correct syntax is `resource_type[resource_name]'" else raise Chef::Exceptions::InvalidResourceSpecification,

rehefiookup(create_key(resource_type, instance_name)) end Chef.deprecated(:multiresource_match, "The resource

else raise ArgumentError, "bad argument to run_context helper, must be :root, :parent, or a Chef::RunContext"



Continuous Compliance in a DevSecOps World

Why Compliance as Code Matters



Presented By



Galen Emery Lead Compliance Architect Chef Software



Adam Montville Chief Product Architect Center for Internet Security

3 chef.io



Toward Compliance Agility Foundational Automation

Adam Montville, Chief Product Architect

2019-10-30



About CIS

- Non-profit using the power of a global community to develop best practices for securing IT systems and data
 - Founded in the year 2000
 - 200+ employees across the U.S.
- Vision: Leading the community to secure our connected world.
- **Mission:** Identify, develop, validate, promote, and sustain best practice solutions for cyber defense. Build and lead communities to enable an environment of trust in cyberspace.

Confidence in the Connected World Start Secure. Stay Secure.®



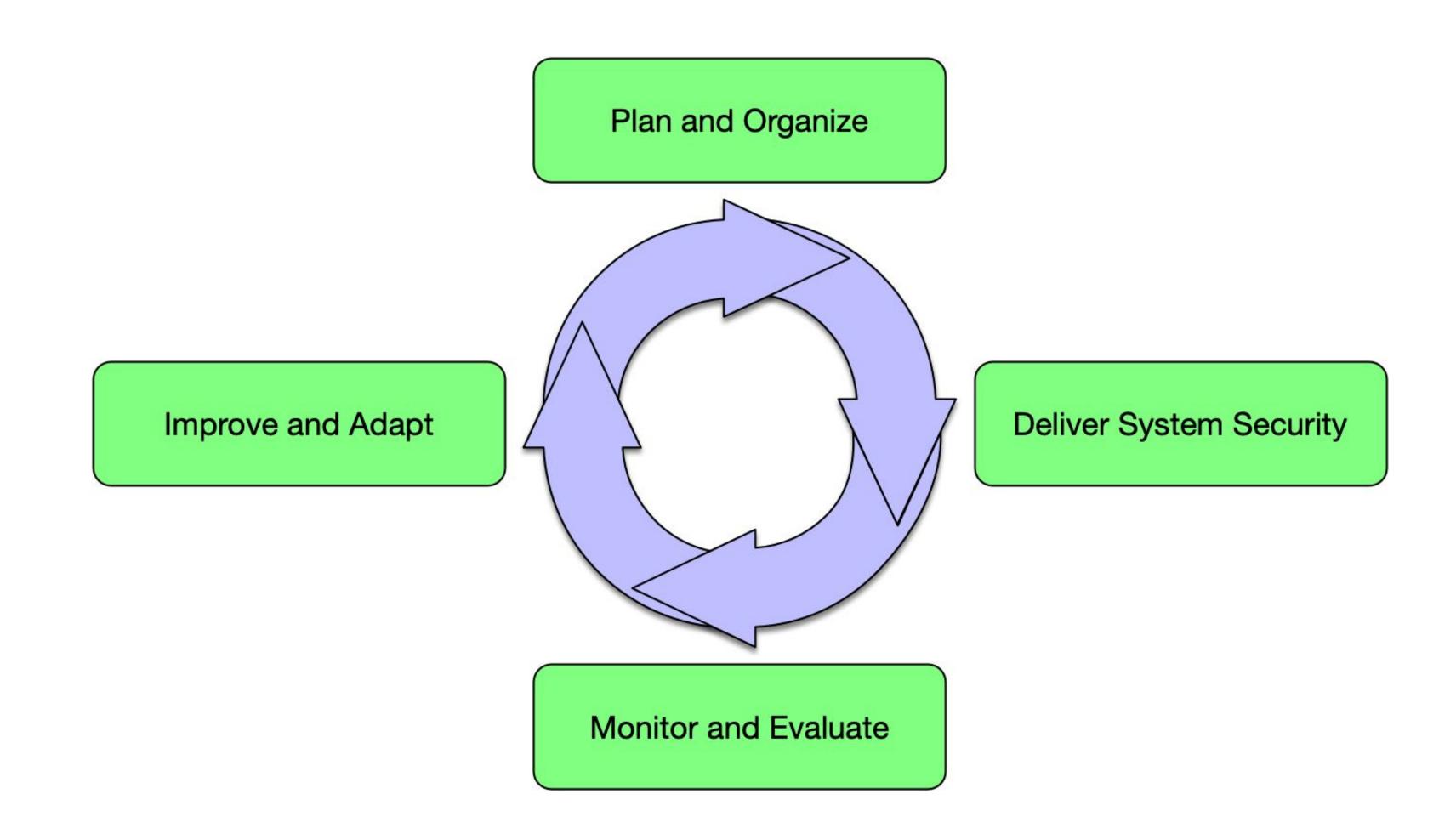
Security Best Practices





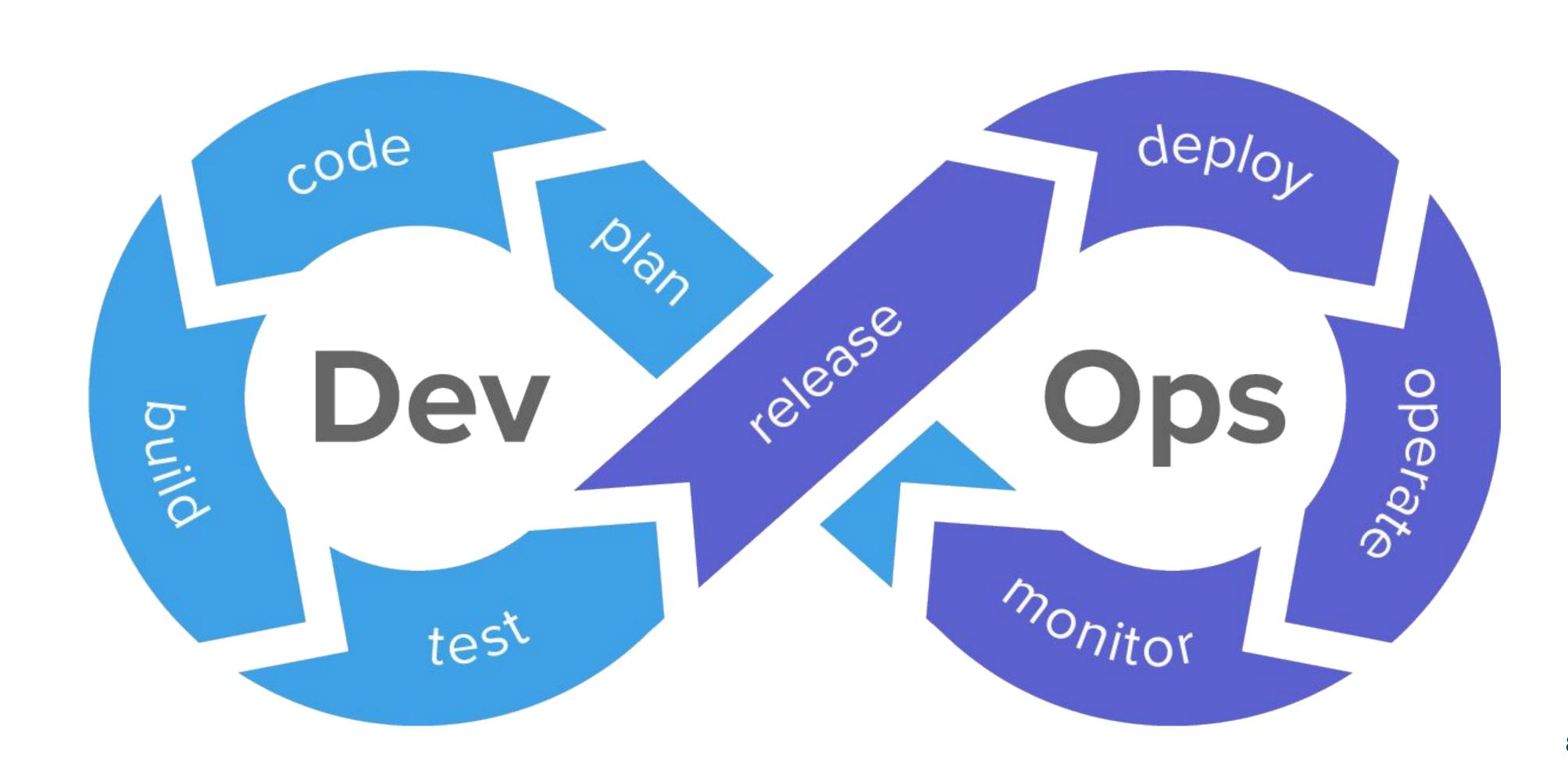


Traditional Compliance Paradigm



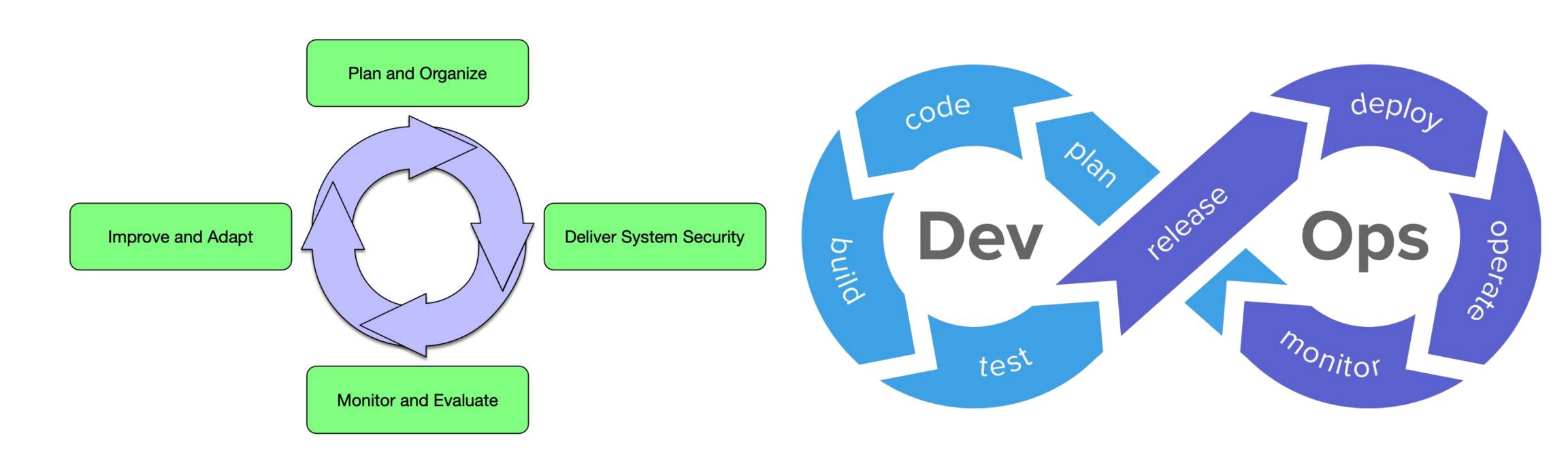


Devops Paradigm





What are the real differences?





How can Security Programs become DevSecOps?

- Articulate security requirements in code-friendly ways
- Empower your product/service teams
- Think iteratively
- Maximize CI/CD



My (Boring?) Parting Thoughts

2010

- Overall Membership: < 300
- Vendor Members: ~10

2019

- Overall Membership: 10,000+
- Vendor Members: 50+

2018

Benchmark Downloads: 1M+

Compliance as Code

postgres = external_services['postgresql'] begin require 'pg' connection =postgresql_connection(postgres) if detail_level == :sparse "run: postgresql: connected OK to # rage.split("\n")[0]}" else log err_STAT001_postgres_failed(postgres, e.message) Kernel.exit! 128 end ensure connection.close if connection end end def postgres, connection) max_conn = connection.exec("SELECT setting FROM pg_settings WHERE name = 'max_connections'")[0]['setting'] total_conn = connection.exec("SELECT sum(numbackends) num FROM pg_stat_database')[

Why does it matter?

new end def keys @resources_by_key.keys end def insert_as(resource_type = nil, instance_name = nil) is_chef_resource!(resource) resource_type ||= resource.resource.a?(Chef::Resource) key = key.to_s res = @resources_by_key[key] unless res raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)"

_by_key.default raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)" end res end def find(*args) results = Array.ne

s::InvalidResourceSpecification, msg end end flat_results = results.flatten flat_results.length == 1 ? flat_results[0] : flat_results end alias_method :resources, :find urceSpecification, "The string `#{query_object}' is not valid for resource collection lookup. Correct syntax is `resource_type[resource_name]'" else raise Chef::Exceptions::InvalidResourceSpecification resources_by_key"] resources = rl.merge(rl) { |k, r| Chef::Resource.from_hash(r) } collection.instance_variable_set(:@resources_by_key, resources) collection end private deficient

instance_names.each do |instance_name| results << lookup(create_key(resource_type, instance_name)) end end results end def find_resource_by_string(arg) begin arg =~ /^.+\[(.+)\]\$/ resource_list = \$1 resource_list.split(",").each do |instance_name| results << lookup(create_key(resource_type, instance_name)) end Chef.deprecated(:multiresource_match_ "The resource_type, instance_name))



Plugging Security Holes

It never ends





The Delicate Balance





Security & Compliance: The Challenge



Security & Compliance are non-negotiable, but too often impede velocity and create inefficiencies.

- Security reviews at the end of the dev process block progress
- Audit time commitments distract staff from high value work
- Limited visibility and collaboration create undue risk

PERCEPTION



Among IT professionals, 81% believe InfoSec policies inhibit agility and speed.¹



Information Security professionals agree that their policies inhibit agility and speed.¹

1—Gartner, How to Seamlessly Integrate Security Into DevOps (2016)

chef.io



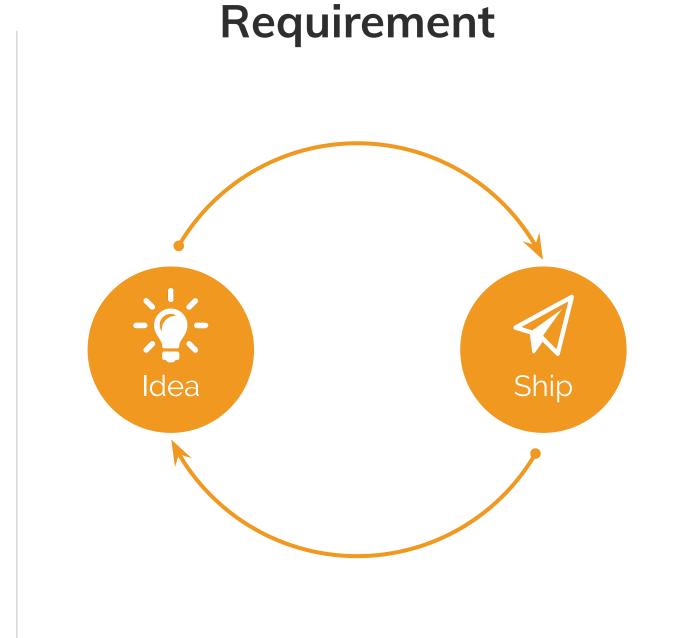
Is Security Slowing Digital Transformation?

Shipping apps and experiences quickly is key to business growth

Problem

Information Security teams aren't set up to act rapidly

- 71% of IT orgs adopt DevOps¹
- DevOps teams work fast
- Continuous delivery of change
- Slow response from InfoSec
- Vulnerabilities and risk



PERCEPTION



Among IT professionals, 81% believe InfoSec policies inhibit agility and speed.¹



Information Security professionals agree that their policies inhibit agility and speed.²

Disrupt or be disrupted. Outperform the competition with digital transformation.

16 chef.io

^{1—}Rightscale, State of the Cloud 2017

^{2—}Gartner, How to Seamlessly Integrate Security Into DevOps (2016)



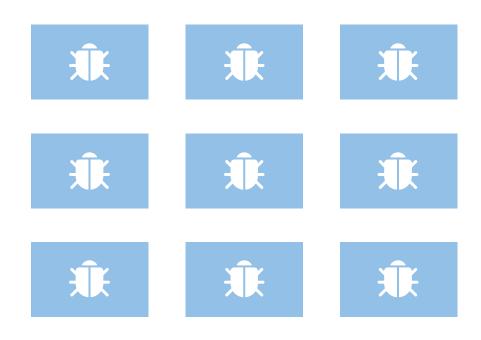
The current state of information security

Despite velocity gains by other teams, InfoSec lags behind

²/₃ of breaches took months or longer to discover ¹



Since 2014, more than 90% of exploits observed use only nine known vulnerabilities



And after a compliance violation or security vulnerability is discovered:



1 in 2 teams need days or longer to remediate, 1 in 4 need weeks or months



30% need days



28%
need weeks or
months

Source: Verizon Data Breach Report 2018

Source: Chef Survey 2017





Traditional Approaches Exacerbate Pain



Security reviews:

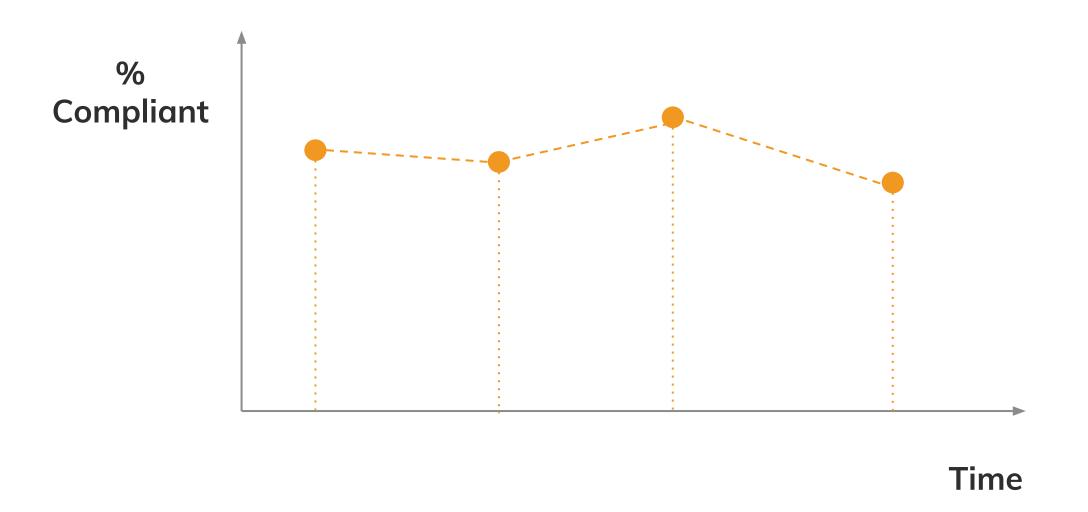
- Are often manual (slow)
- Rely on scanning tools that generate too much data to effectively manage
- Catch problems too late in the development cycle to economically fix
- Don't manage exceptions appropriately



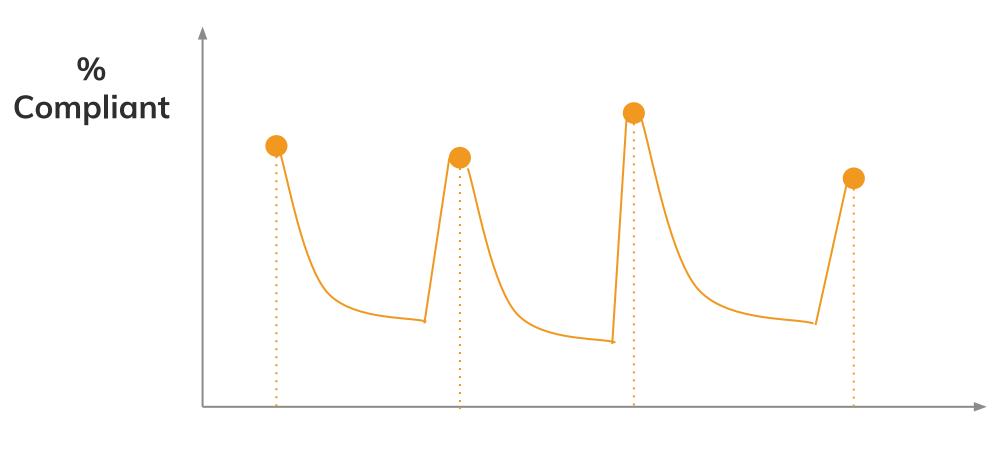
Periodic Audits Do Not Tell the Whole Story

They leave open windows of risk

Assumed Compliance Levels



Actual Compliance Levels



Time



Different Teams Speak Different Languages







20 chef.io



Bridging the Gap - Security Meets Operations

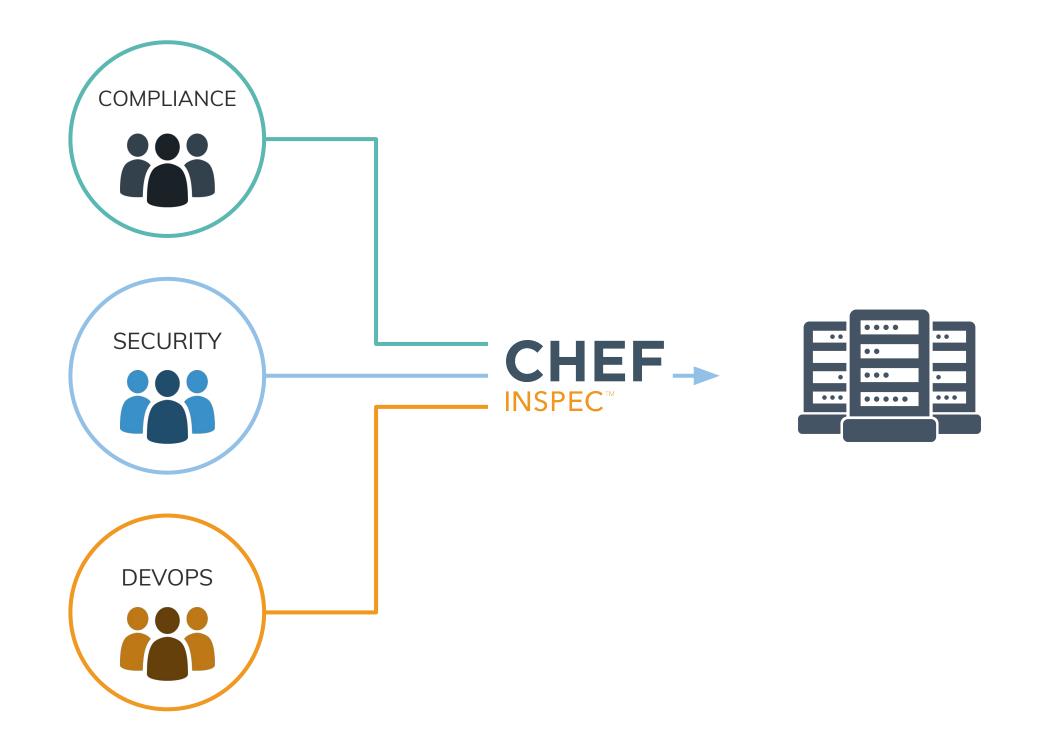
The Old Way
People working directly on machines



The Modern Way
People automating machines using code



The New Way
Shared tooling across organizations



1 chef.io



Manage Security and Compliance as Code

Code enables continuous compliance

Collaborate

Code is an unambiguous common language

Enable scalability

Code scales across complexity sprawl

Shift left

Test throughout the software delivery process

Continuous visibility

Monitor ongoing basis to eliminate windows of risk

```
control 'ssh-1234' do
  impact 1.0
  title 'Server: Set protocol version to SSHv2'
  desc "
    Set the SSH protocol version to 2. Don't use
legacy
    insecure SSHv1 connections anymore...
"

describe sshd_config do
  its('Protocol') { should cmp 2 }
  end
end
```



Continuous Compliance as Code



CHEF INSPEC Continuous compliance



Customer Evidence - Compliance Automation



Every resource and app in HPC environment is automatically qualified as compliant with FDA standards before deployment; previously this was only checked four times per year

■ SCHOLASTIC

Remediated
Shellshock security
issues across entire
cloud infrastructure in
a single hour, a
process that took more
than a day for its
smaller non-cloud
infrastructure

Web & Media Giant

Can patch entire infrastructure of 250,000 nodes within 6 hours of a patch being made available

Top 5
Global Bank

Eliminated \$2M in security and network management software toolset costs while reducing remediation time across thousands of servers from weeks to one hour



CIS Security Software Certification for CIS Benchmark(s)

- CIS Benchmark for Amazon Web Services Foundations Benchmarks, Level 1
- CIS Benchmark for Amazon Web Services Foundations Benchmarks, Level 2
- CIS Benchmark for Google Cloud Platform Foundation, Level 1 Profile
- CIS Benchmark for Google Cloud Platform Foundation, Level 2 Profile
- CIS Benchmark for CIS Red Hat Enterprise Linux 7 Level 1 Server
- CIS Benchmark for CIS Red Hat Enterprise Linux 7 Level 2 Server
- CIS Benchmark for Microsoft Azure Foundations Level 1



First CIS Partner Certified on AWS, Azure, and GCP

CHEF

| Superior of the content of t

res, connection) — max_conn = connection.exec("SELEC) setting FROM pg_stat_database')

tgreSQL * Connected to PostgreSQL * Connections: * Processes ids pending locks: EOM — end — def err_STAT001_postgres_failed(postgres, message)<<EOMSTAT001: An error occurred while attempting to get status from PostgreSQL server at #{postgres['vip']}. The error reported was:#{format_multiline_message(16, message) to the file below: #{past}}FOM — msg << <<FOM See https://docs.chef.io/error messages.html#cleanse001-postgres-failed for more information FOM — msg — end — def warn CLEANSE002 no postgres delete(sql)

oscode-solr4: failed to connect to #{solr}: #{e.message.split("\n")[0]}" end end def external_cleanse_opscode_solr4(perform_delete) log <<-EOMCleansing data in a remote Sol4 instance is not currently solf status.success? log "#{display_name} Reconfigured!" exit! 0 else exit! 1 end end endendrequire 'pathname'file_path = Pathname new(cmd_name, true, "Chef Server")ChefServerCtl::Config.init(ctl)require "chef/resource_collection/resource_collection_serialization"class Chef class ResourceCollection class ResourceSet

a?(Chef::Resource) key = key.to_s res = @resources_by_key[key] unless res raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)"

_by_key.default raise Chef::Exceptions::ResourceNotFound, "Cannot find a resource matching #{key} (did you define it first?)" end res end def find(*args) results = Array.new
ons::InvalidResourceSpecification, msg end end flat_results = results.flatten flat_results.length == 1 ? flat_results[0] : flat_results end alias_method :resources, :find
sourceSpecification, "The string `#{query_object}' is not valid for resource collection lookup. Correct syntax is `resource_type[resource_name]'" else raise Chef::Exceptions::InvalidResourceSpecification,

"@resources_by_key"] resources = rl.merge(rl) { |k, r| Chef::Resource.from_hash(r) } collection.instance_variable_set(:@resources_by_key, resources) collection end private def creater instance_names.each do |instance_name| results < lookup(create_key(resource_type, instance_name)) end def find_resource_by_string(arg) begin stance = \frac{1}{2} \text{ | |k, r| Chef::Resource_type, instance_name| end end private def creater | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) begin | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) begin | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_by_string(arg) | |k, r| Chef::Resource_type, instance_name | end def find_resource_type, instance_n