

DevOps: Crafting Reusable Infrastructure Resources

Jennifer Davis

Network Name: WPOC_WiFi

Access Code: velocity2016

Jennifer Davis

- Software Engineer, Chef Software
- Co-author of "Effective Devops"
- Founder of CoffeeOps
- DevOpsDays Silicon Valley Organizer

O'REILLY®



Effective DevOps

BUILDING A CULTURE OF COLLABORATION,
AFFINITY, AND TOOLING AT SCALE

Jennifer Davis & Katherine Daniels

Katie Rose

Jonathan Disher

Ian Henry

Communication

- Jennifer Davis
Twitter: @sigje #effectivedevops
Email: sparklydevops@gmail.com
- Katie Rose Twitter: @RealKatieRose
- Jonathan Disher Twitter: @funjon
- Ian Henry Twitter: @Eeyun__

Feedback

- Constructive feedback
 - What did you find helpful?
 - What would you like to see more/less of?
 - Was there anything you found unclear?

Schedule

Morning Break: 10:30 - 11:00am

Lunch: 12:30 - 1:30pm

Afternoon Break: 3:00 - 3:30pm

Network Connectivity

Network Name: WPOC_WiFi

Access Code: velocity2016

Expectations

- Safe space to share experiences, learn from each other
- Code of Conduct
- Learn effective workflows for using and testing source control and configuration management

Team Activity 1

- Meet your team!
 - What are motivations?
 - What are current beliefs?
 - Skills? Gaps in skills?
 - git, chef, docker, continuous integration, continuous delivery

<http://goo.gl/forms/M2bvntFSYqxtnlzb2>

What is Devops

What is Devops

Cultural movement that seeks to:

- change how individuals work,
- value the diversity of work done,
- develop conscious decisions in acceleration,
- plan for scale, and
- measure the effect of social and technical change.

Folk Models

- general popularly understood meaning particular to a socio-cultural grouping but which has not been formally defined or standardized.

Why Devops?

**High Performing Devops Teams
are more agile
30X more frequent deployments
8000X faster lead times than peers**

2014 PuppetLabs State of DevOps Survey

**High Performing Devops Teams
are more reliable
2X change success rate
12X faster mean time to recovery (MTTR)**

2014 PuppetLabs State of DevOps Survey

Foundation of Devops

Collaboration

Individuals working together with shared interactions and input building towards a common goal.

Collaboration

- Multiple types of collaboration
- Monitoring for single points of failure
- Monitoring burnout

Smarter Teams build better value

Affinity

Building inter-team relationships, empathy and trust in support of shared organizational and business goals.

Affinity

- Monitor organizational signal
- Monitor gating of processes
 - Shadow HR, IT, Marketing

Tools

Accelerators of culture that if used effectively can enhance and support a culture of collaboration and affinity.

Scaling

Applying the considerations of collaboration, and tooling throughout various inflection points of an organizations lifecycle.

Foundations of DevOps

- Collaboration
- Affinity
- Tools
- Scaling

Framing Devops Narrative

The Devops Compact

- shared mutual understanding
- established boundaries

Team

- Common purpose
- Defined beliefs
- Empowered

Diversity in Teams

- Professional
- Personal
- Goals
- Cognitive Styles

Team Activity 2

Careless Conversations (inspired by Alan Cyment)

- Pair up
- Select one person to go first.
- For one minute, speaker talks about something passionate about. Listener stays seated, quiet, and acts disinterested.
- Switch roles and repeat.
- Repeat until both people have done this twice.

Careless Conversations

- How does it feel not to be listened to?
- How does it feel to ignore someone?

Cultivating Empathy

- Collect stories
- Listen
- Circle back



**Shared stories
articulate values and ideas.**



**Shared stories
explain historical significance.**

A dense crowd of people in a painting by Gustave Caillebotte, 'The Parisians on the Quai Voltaire'. The scene depicts a busy street with many figures, some in hats and coats, others in more casual attire, all appearing to be in motion or engaged in conversation.

Shared stories passing knowledge

**Shared stories
influence community.**

**Shared stories
cultivate empathy.**

A life becomes meaningful when one sees himself or herself as an actor within the context of a story.

-- George Howard

Mindsets

- Fixed
- Growth

Small vs Large teams

- Large teams - roles may be highly segregated
- Small teams - one person may be responsible for many roles

Critical Habits for Teams

- Code Review
- Pairing

Code Review

- Max 90 minutes in one setting

Pairing

- Agile software development
- 2 people work together on 1 workstation
- Driver - writes code
- Observer - reviews each line
- Roles switch frequently

Types of Pairing

- Expert-expert
- Expert-novice
- Novice-novice

GridIronOps - Katie Rose

Team Activity 2.5:

Coat of Arms

Visualizing Work

- Break down rigid single points of knowledge failure
- Reduce Development friction
- Eliminate duplicate efforts

A solution

- Team kanban board
- Commitments to incremental improvements

Factors for Success

- management buy in
 - training
 - effort to minimize "pushing"
- weekly team syncs
- proximity of team

Prepping for Success

- Environment
- Values
- Desire
- Motivation
- Connectedness

Team versus Individual

Objectives

- Defined by the team. Not management.
- Defined by the team. Not individuals.
- Everyone has voice, opportunity to speak.

As a team...

- Discuss objectives.
- Describe work.
- Define lanes.
- Define a task.
- Define a project.

Elect a champion

Workflow

Work that is

- orchestrated
- repeatable
- organized
- moves from one state to another

WIP

- Work in Progress
 - work that has had money or people applied to it.

Work Identification

- name
- start date
- end date
- current state
- description
- priority

Task Handling

- What is it?
- Can you do anything with it?
- What is the next step?

Projects

- Same requirements as a task
- Larger in scope
- May be comprised of more than one task

Interrupts

- Non planned work that comes in
 - customer request
 - incident
 - request for help from coworker
 - single point of knowledge (you) work
 - high priority task push from manager

Blocked work

- Work that can progress no further:
 - dependent teams - blocked by external team
 - insufficiently qualified request - blocked by requester
 - dependent on SPOK - blocked by team
 - time dependent

Team Activity 3

Discuss with your team:

- What is the difference between a task and a project?
- Do you have interrupts? What are they?
- How do you determine when work is done?

Time: 15 minutes

Kanban is a system.

- Visual process management
 - what to do
 - when to do it
 - how much to do

Kanban is a method.

- incremental, evolutionary process improvement

3 principles

- Current Process
- Incremental, evolutionary change
- Respect

Current Process

- Do you know what the current process is?
- Is it documented? Is it explicit? Is it clear?
- Has it been evaluated with the team?

Incremental Evolutionary Change

- How are you measuring current process?
- Is the value clearly understood?
 - Is work defined in value or cost?

Respect

- Find the current value.
- Not forceful in nature.
- What people want versus how to get there

Kanban

- Start with what you do now
- Agree to incremental, evolutionary change
- Respect
- Everyone is a leader

Kanban Practices

- Visualize
- Limit WIP
- Manage flow
- Make policies explicit
- Implement feedback loops

Visualize

- Intent
- Alignment
- Coherence

Limit WIP

- Pull (don't push)

Manage Flow

- Monitor/measure/report
- Incremental change

Make Policies Explicit

- Document processes
- Group signoff

Implement Feedback Loops

- Collaboration
- Retrospectives

Team Activity 4

Talk through workflow for tasks for your team. How will you figure out what work needs to be done, who will work on the work, and when it is done. Use postits to mock up a legend for types of work items. Use pad to mock up your work items board. Use blue tape to mark off lanes.

Time: 30 minutes

Affinity

Devops Tools

- Establish local development environment
- Version control
- Manual -> Automation -> Continuous
 - Artifacts
 - Infrastructure
 - Sandbox

Local Development Environment (LDE)

- Consistent set of tools across the team
- Ability to quickly onboard new engineers

Provisioned Node - LDE

- AWS instance node
- Chef DK
 - Test Kitchen
 - Ruby
 - ChefSpec, ServerSpec
- Git

Configuration Management

- Process of identifying, managing, monitoring, and auditing a product through its entire life including the processes, documentation, people, tools, software, and systems.

Version Control

- Records changes to files or sets of files stored within the system
- Enable revisions
- Integrity checking
- Collaboration

Artifact Repository

- Secure
 - Trusted
 - Stable
 - Accessible
 - Versioned
- (artifactory, nexus, yum, package.io, rubygems)

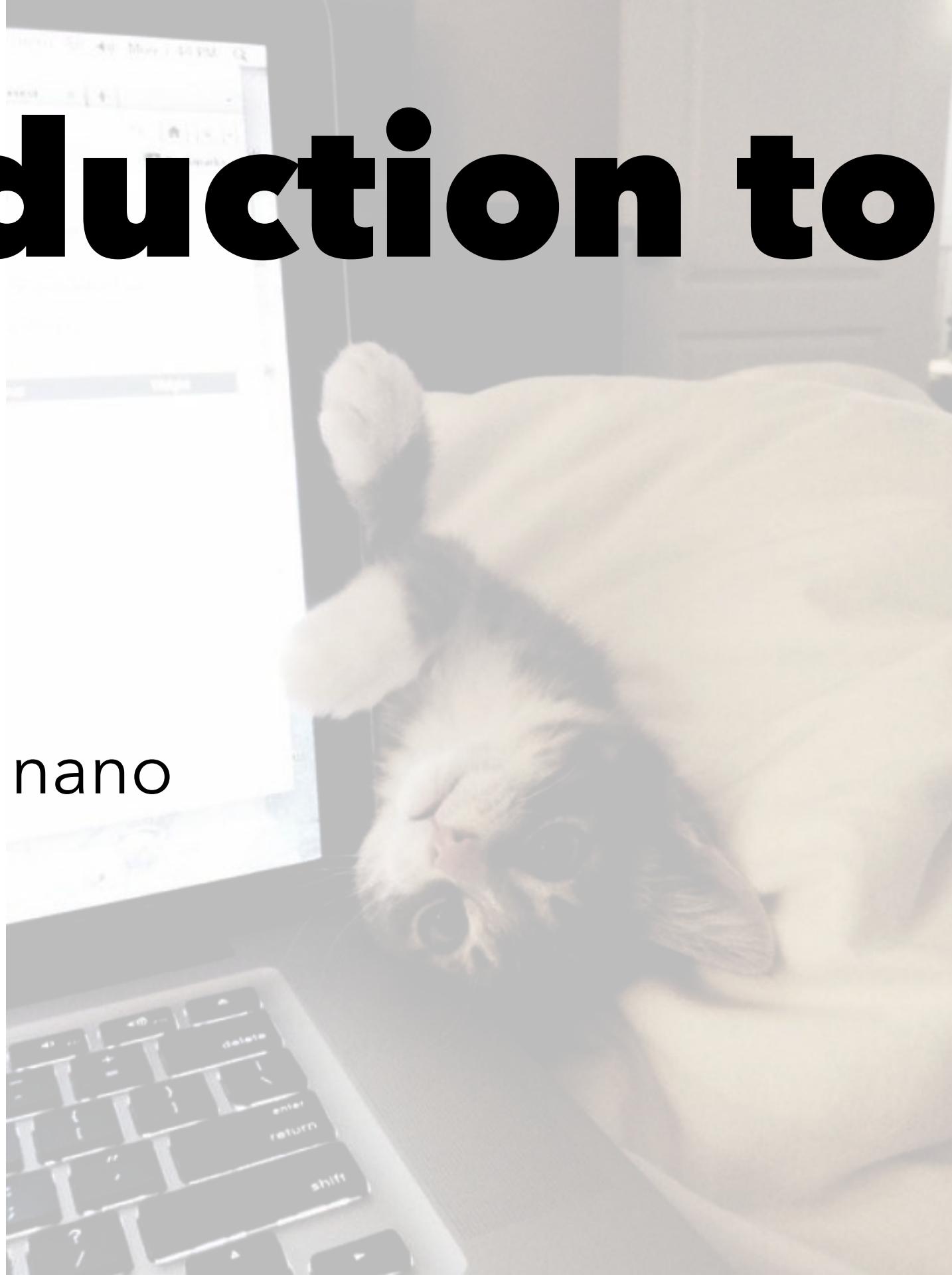
Introduction to Lab 1

Lab 1

User: chef

Password: chef

sudo yum install nano



Lab 1

Lab 1

Time: 15 minutes

Team Activity 5

In pairs, discuss your current work environment.

- Who are the members of your team (at work)?
- Who are the people who have commit access?
- What's the flow of code from design to deploy?

While one person shares their environment, the other person should draw a diagram to represent the information shared. Use circles to represent people, triangles to represent code

Introduction to Lab 2

Lab 2



Lab 2

Lab 2

Time: 15 minutes

Extending Git Understanding

- Highly recommend: Git for Teams <http://gitforteams.com/>
Emma Jane Westby

Infrastructure

- Aggregate of applications, configurations, access control, data, compute nodes, network, storage, processes, and people.

Infrastructure Automation

- Systems that reduce the burden on people to manage services and increase the quality, accuracy and precision of a service to the consumers of a service

Infrastructure Automation Tools

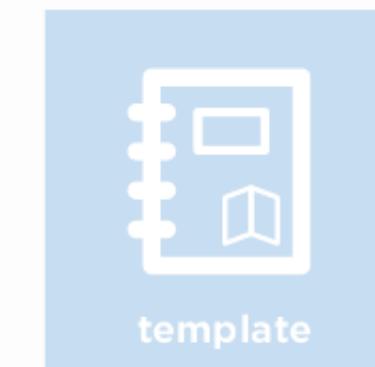
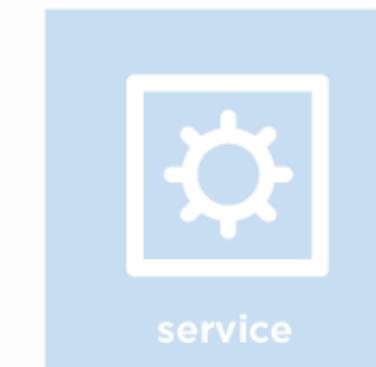
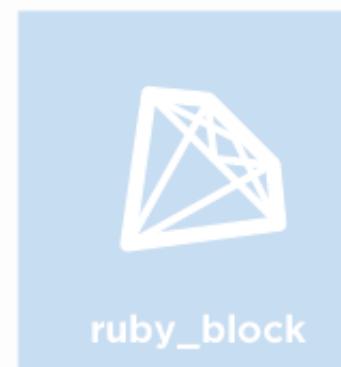
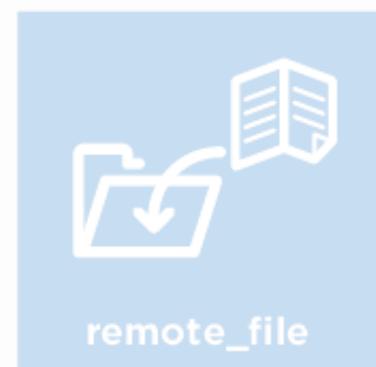
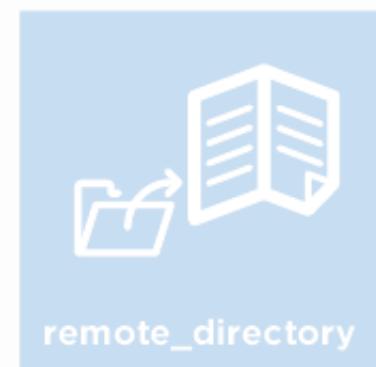
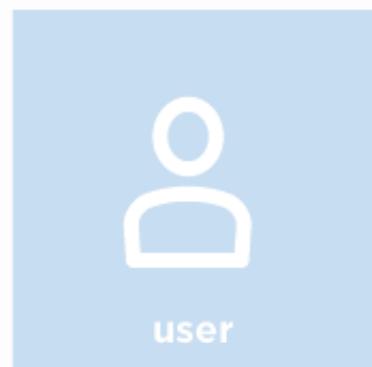
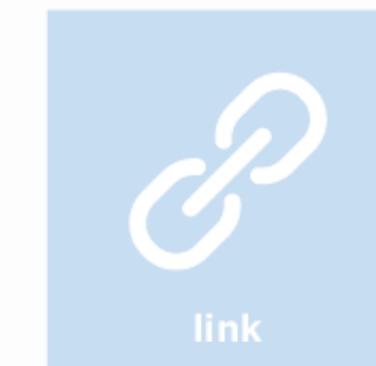
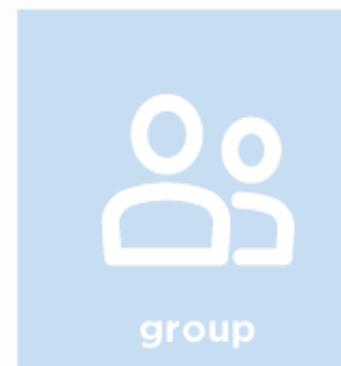
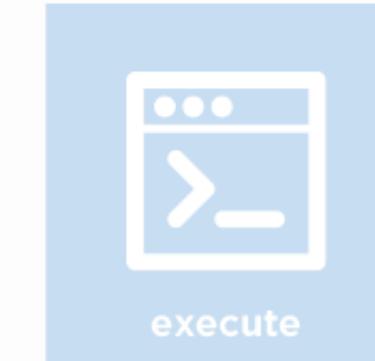
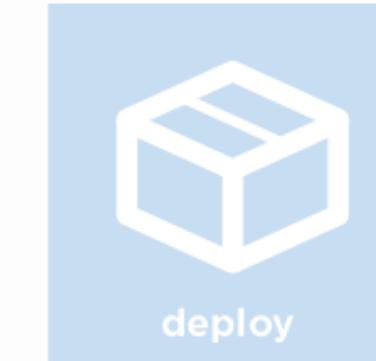
- Chef
- Puppet
- Ansible
- Salt
- CFEengine

Introduction to Chef

, Baking Cookies

Resources

- Ingredients of infrastructure



Resource Declaration

```
RESOURCETYPE "RESOURCE_NAME" do  
    PARAMETER PARAMETER_VALUE  
end
```

Example Resource Type - package

A package to be installed:

```
package "httpd" do
  action :install
end
```

Example Resource Type - service

A service that should be started:

```
service "httpd" do
  supports :restart => :true
  action [:enable, :start]
end
```

Resources

A resource is a statement of policy that:

- Describes the desired state for an element
- Specifies a resource type---such as package, template, or service
- Lists additional details (also known as parameters), as necessary
- Are grouped into recipes

Recipes

- Collection of ordered resources
- Combination of ruby and Chef DSL

Cookbooks

- Thematic
- Collection of recipes and other supporting files

Roles

- Abstraction describing function of system
- Name
- Description
- Run list (ordered list of recipes and roles)

Run List

- Ordered list of recipes and roles
- Specific to a node

Nodes

- Machine (virtual, physical, cloud server, or other device) that is managed by Chef

Environments

- Abstraction models workflow
- Name
- Description
- Cookbook version pinning

Supermarket

- Community site with a number of cookbooks
- Read before using in your environment

Chef DK

- Chef development kit
- Includes a number of utilities and software to facilitate cookbook creation
- Free download off of the website

Berkshelf

- Dependency management
- Included with Chef DK

Review of Day 1

Schedule

Morning Break: 10:30 - 11:00am

Lunch: 12:30 - 1:30pm

Afternoon Break: 3:00 - 3:30pm

Foundations of DevOps

Foundations of DevOps

- Collaboration
- Affinity
- Tools
- Scaling

Collaboration

Individuals working together with shared interactions and input building towards a common goal.

Affinity

Building inter-team relationships, empathy and trust in support of shared organizational and business goals.

Tools

Accelerators of culture that if used effectively can enhance and support a culture of collaboration and affinity.

Scaling

Applying the considerations of collaboration, and tooling throughout various inflection points of an organizations lifecycle.

Collaboration

- Shared stories
- Cultivating Empathy
- Diversity in teams

Affinity

Tools

- Local development environment - Chef DK
- Version Control - git
- Infrastructure Automation - chef
- Sandbox Automation - Test Kitchen

Test Kitchen

- Included with Chef DK
- Sandbox automation
- Test harness

Test Kitchen

- Execute code on one or more platforms
- Driver plugins supporting various cloud and virtualization providers

.kitchen.yml

- driver
- provisioner
- platforms
- suites

.kitchen.yml driver

- virtualization or cloud provider

Example: vagrant, docker

.kitchen.yml provisioner

- application to configure the node

Example: chef_zero

.kitchen.yml platforms

- target operating systems

Example: centos-6.5

.kitchen.yml suites

- target configurations

Example:

```
name: default
  run_list:
    - recipe[apache::default]
  attributes:
```

Kitchen commands (1/2)

- kitchen init
- kitchen list
- kitchen create
- kitchen converge

Kitchen commands (2/2)

- kitchen verify
- kitchen destroy
- kitchen test

Introduction to Lab 3

Lab 3

Lab 3

Time: 30 minutes

Runbook to Recipes

Introducing MongoDB

MongoDB terminology

- document - basic unit of data for MongoDB, ordered set of keys with associated values
- collection - group of documents. analogous to table.
- instance - MongoDB can host multiple independent databases, each of which can have its own collections.
- `_id` - special key unique within a collection.

MongoDB Shell

- mongo
- full featured JavaScript interpreter

Introduction to Lab 4

Translate a runbook for installing MongoDB into chef.

Lab 4

Lab 4

Time: 30 minutes

Introduction to Lab 5

Translate our MongoDB cookbook from recipes into resources.

Lab 5

Lab 5

Time: 30 minutes

Managing Risk

- Test
- Small frequent releases

Linting

- Ensure code adheres to styles and conventions
- Weave expectations into development
- Encourages collaboration

Testing

- Documenting objectives and intent
- Measuring "done"

Code Correctness

- foodcritic
- rubocop

Integration Tests

- ServerSpec
- InSpec

Rubocop

- Ruby linter
- Analyzes source code to verify the syntax and structure of our cookbooks.
- [Ruby style guide](#)
- Included with ChefDK

Why

- identify simple errors early
- improve the review process by eliminating stylistic issues

Rubocop Example

```
$ rubocop cookbooks/COOKBOOK1 cookbooks/COOKBOOK2 cookbooks/COOKBOOK4
```

Reading Rubocop Output

Inspecting 8 files
CWCWCCCC

- . means that the file contains no issues
- C means a issue with convention
- W means a warning
- E means an error
- F means an fatal error

Rubocop Cops

- Lint
- Rails
- Style

Example: rubocop --lint.

Disabling Rubocop cops

Any configuration in `.rubocop.yml` is disabled.

To disable string literals:

```
StringLiterals:  
  Enabled: false
```

Foodcritic

- Chef linter
- Chef style guide
- Included with ChefDK

Foodcritic Example

```
$ foodcritic cookbooks/setup
```

Reading Foodcritic Output

FC008: Generated cookbook metadata needs updating: ./metadata.rb:2

ServerSpec

- Tests to verify servers functionality
- Resource types
 - Package, service, user, and many others
- Integrates with Test Kitchen
- <http://serverspec.org>

ServerSpec Generic Form

```
describe "<subject>" do
  it "<description>" do
    expect(thing).to eq result
  end
end
```

ServerSpec Potential Tests

- Is the service running?
- Is the port accessible?
- Is the expected content being served?

ServerSpec Example

```
describe 'apache' do
  it "is installed" do
    expect(package 'httpd').to be_installed
  end
  it "is running" do
    expect(service 'httpd').to be_running
  end
end
```

Reading ServerSpec Output

```
app::default
  httpd service is running
```

```
Finished in 0.26429 seconds (files took 0.7166 seconds to load)
1 example, 0 failures
```

InSpec

- infrastructure specification
- open-source testing framework for infrastructure
- human and machine readable
- compliance, security, and policy requirements
- Based off of ServerSpec

InSpec Example

```
describe service 'ssh-agent' do
  it { should be_running }
end
```

InSpec Example Run

```
$ inspec exec test.rb
```

.

```
Finished in 0.00901 seconds (files took 0.98501 seconds to load)
1 example, 0 failures
```

InSpec Remote Run

```
$ inspec exec test.rb -i ~/.aws/YOUR.pem -t ssh://ec2-user@54.152.7.203
```

InSpec Remote Run on Windows

```
$ inspec exec test.rb -t winrm://Admin@192.168.1.2 --password super
```

InSpec Docker Containers

```
$ inspec exec test.rb -t docker://3dda08e75838
```

More info on InSpec

https://github.com/chef/inspec/blob/master/docs/ctl_inspec.rst

InSpec Audit Rules

- lots of resources to facilitate auditing nodes

InSpec host resource example

```
describe host('example.com', port: 80, proto: 'tcp') do
  it { should be_reachable }
end
```

InSpec login_defs resource example

```
describe login_defs do
  its('PASS_MAX_DAYS') { should eq '180' }
  its('PASS_MIN_DAYS') { should eq '1' }
  its('PASS_MIN_LEN') { should eq '15' }
  its('PASS_WARN_AGE') { should eq '30' }
end
```

InSpec Reference

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

Introduction to Lab 6

Lab 6

Lab 6

Time: 10 minutes

Introduction to Lab 7

Lab 7

Lab 7

Time: 20 minutes

Test, Monitor, or Diagnostic²

1. Where is it going to run?
2. When is it going to run?
3. How often will it run?
4. Who is going to consume the result?
5. What is the entity going to do with it?

² Lam, Yvonne. 'Sysadvent: Day 5 - How To Talk About Monitors, Tests, And Diagnostics'. Sysadvent.blogspot.com. N.p., 2014. Web. 26 May 2015.

Intro to Jenkins

- continuous integration server to build and test software.
- integrated with source control (svn, git, ..)
- integration with build systems
- worker nodes to scale out systems
- plugins (complex and community maintained)

Layers of Infrastructure Management

- Orchestration
 - Coordination
 - Scheduling
- Configuration packages and software
- Provisioning

Jenkins terminology

- job - base unit of work.

General workflow

- source control change
- jenkins polls source control, triggers a build
- job runs, reports failure or success

Difference between jenkins and chef

- jenkins generally for build and test
- chef for automation of configuration, node deploy to comply to policy

Jenkins can deploy! Jenkins as coordinators.

Closer comparison

Chef delivery versus Jenkins!

Introduction to Lab 8

Lab 8

Lab 8

Time: 30 minutes

Introduction to Lab 9

Lab 9

Lab 9

Time: 20 minutes

Measuring Impact and Value of Change

Impact of Change

Impact on Availability

- Overall site/app availability
- Individual service availability

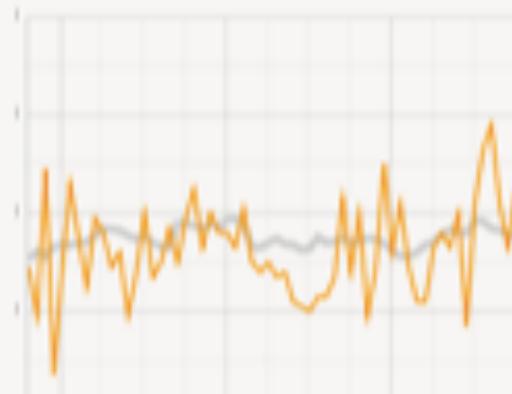
Availability Monitoring

- Uptime:
- Pingdom, Monitis, Uptrends, etc
- Vertical Line Technology:
- Availability after deploys/changes

End-User (1 hour) ☰

Three-Armed Sweaters ⚡ ⓘ 🔎

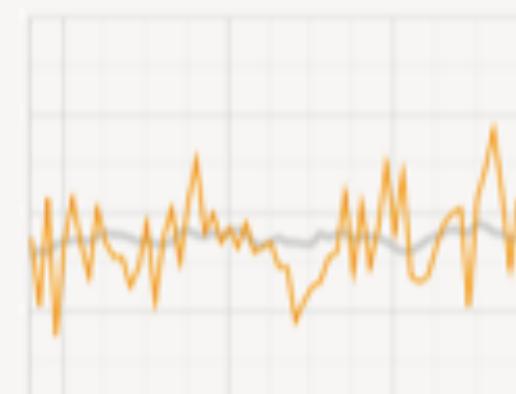
ages · historical



17:00 17:20 17:40

Screwed Users ⚡ ⓘ 🔎

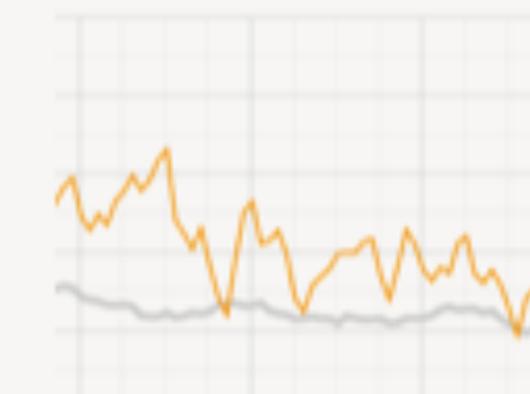
unique users · historical



17:00 17:20 17:40

HTTP 404 (0.27%) ⚡ ⓘ 🔎

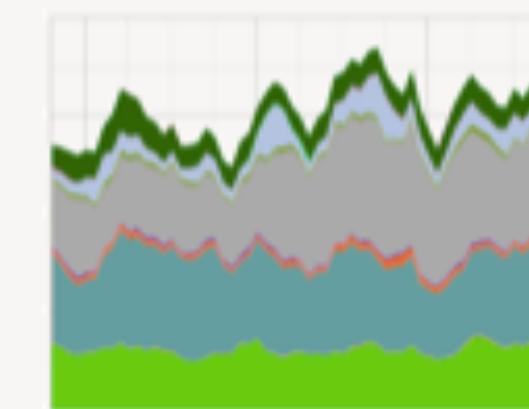
current · historical



17:00 17:20 17:40

HTTP Errors (0.25%) ⚡ ⓘ 🔎

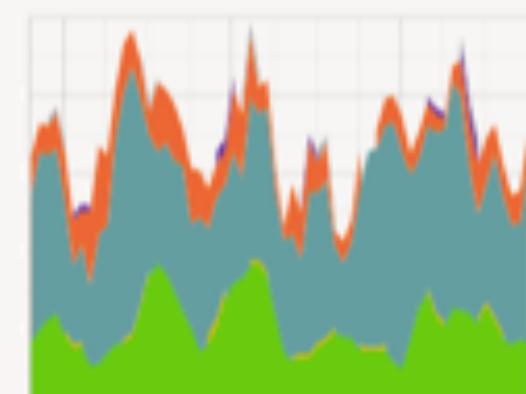
201 · 202 · 206 · 303 · 304 · 400 · 401 ·
403 · 405 · 409 · 410 · 416 · 429 · 439 ·
500 · 501



17:00 17:20 17:40

Native Apps HTTP Errors ⚡ ⓘ 🔎

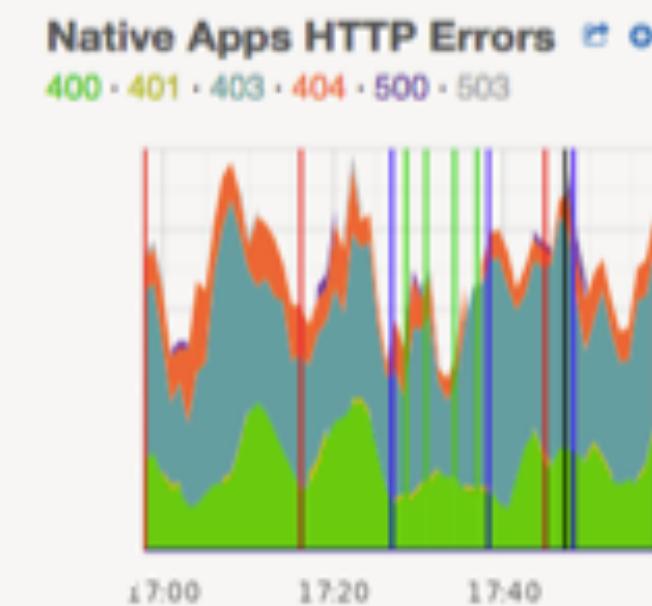
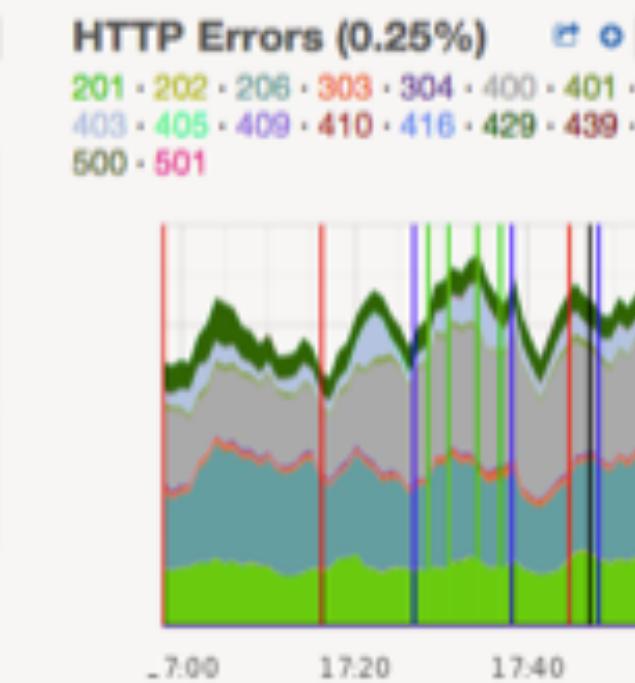
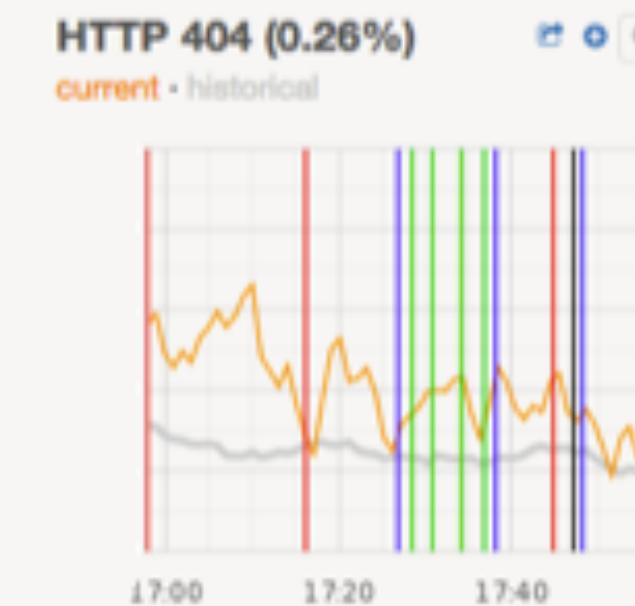
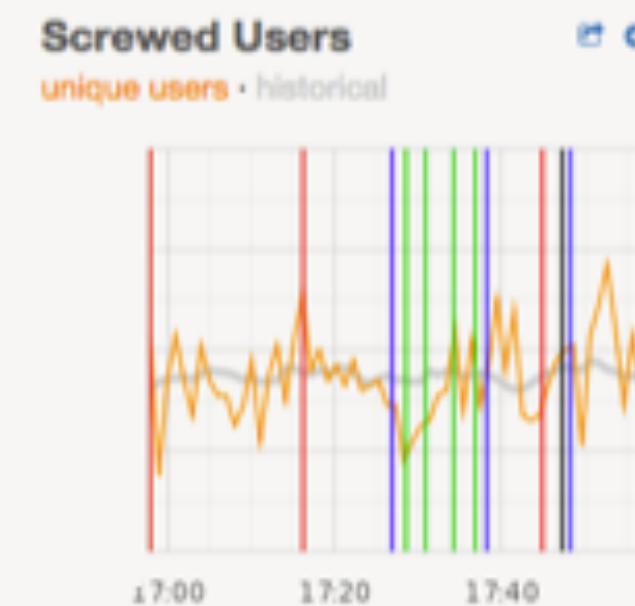
400 · 401 · 403 · 404 · 500 · 503



17:00 17:20 17:40

Eventinator

End-User (1 hour) ■



Service Availability

- Nagios: Service-level monitoring and alerting
- Nagios-herald: Alert context
- OpsWeekly: Historical alert data

Nagios

```
define command {
    command_name      check_mongodb_query
    command_line      $USER1$/nagios-plugin-mongodb/check_mongodb.py
                      -H $HOSTADDRESS$ -A $ARG1$ -P $ARG2$
                      -W $ARG3$ -C $ARG4$ -q $ARG5$
}

define service {
    use                  generic-service
    hostgroup_name       Mongo Servers
    service_description  Mongo Connect Check
    check_command        check_mongodb!connect!27017!2!4
}
```

```
define servicedependency{
    host_name                               WWW1
    service_description                     Apache Web Server
    dependent_host_name                   WWW1
    dependent_service_description        Main Web Site
    execution_failure_criteria          n
    notification_failure_criteria      w,u,c
}
```

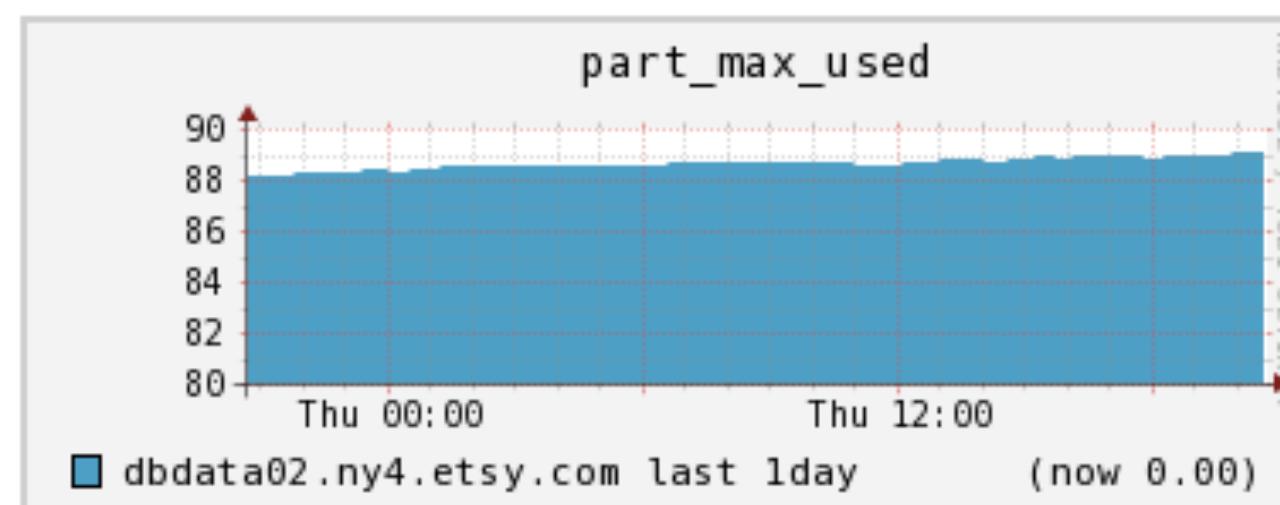
Nagios-herald

nagios@etsy.com

to me ▾

Host: dbdata02.ny4 Service: Disk Space

State is now: **WARNING** for 0d 0h 2m 53s (was WARNING) after 3 / 3 checks



```
THRESHOLDS - WARNING:10%;CRITICAL:5%;  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/cciss/c0d0p2    29G  2.4G  25G  9%  /  
/dev/cciss/c0d0p4  516G  459G  57G  90%  /var  Free disk space (10%) is <= WARNING threshold ( 10%).  
/dev/cciss/c0d0p1   190M   12M  169M  7%  /boot  
tmpfs            12G     0   12G  0%  /dev/shm
```

Alert Frequency

✨ @sigje #effectivedevops ✨

HOST 'dbdata02.ny4' has experienced 1 WARNING alerts for SERVICE 'Disk Space' in the last 7 days.

Problematic volume

Ganglia graph

Threshold exceeded

Alert frequency

opsWeekly

Date/Time	Host	Service	Output	State
Fri 13 Jun 14:04:29 EDT	logarchive02	Disk Space	DISK WARNING - free space: /logs 6616027 MB (10% inode=99%):	WARNING
			No action taken: Threshold adjustmer	
			Notes: Threshold is too low, increased	<input type="checkbox"/>
Fri 13 Jun 04:51:31 EDT	virt14	Disk Space	DISK CRITICAL - /var is not accessible: Input/output error	CRITICAL
			Action taken: Service Issue (View cle)	
			Notes: RAID failure caused machine to die	<input type="checkbox"/>
Thu 12 Jun 20:36:19 EDT	localhost	Aggregate MySQL Slave	OK=46 WARNING=0 CRITICAL=1 UNKNOWN=0 services=/^MySQL Slave/ hosts=/^db(shard	CRITICAL
			Action taken: Service Issue (View cle)	
			Notes: MySQL slave failure on host	<input type="checkbox"/>
Thu 12 Jun 19:12:43 EDT	database001b	Host Check	(Host Check Timed Out)	DOWN
			Action taken: Service Issue (View cle)	
			Notes: Machine died, hardware failure	<input type="checkbox"/>
Wed 11 Jun 12:57:21 EDT	api05	Memory	CHECK_NRPE: Socket timeout after 30 seconds.	UNKNOWN

Ops Weekly Updates - Rep x

opsweekly.etsycorp.com/report.php

Opsweekly Overview + Add Reports Meeting Search Reports and Notifications Set Date Timezone Idleness

On Call Reporting

Week Year

Week Stats

for week Friday 16th May 2014 - Friday 23rd May 2014

60 notifications received this week

Alert Status Distribution

Breakdown of the type of notifications received during the week



CRITICAL	29 (48.33%)
WARNING	20 (33.33%)
DOWN	7 (11.67%)
UNKNOWN	4 (6.67%)

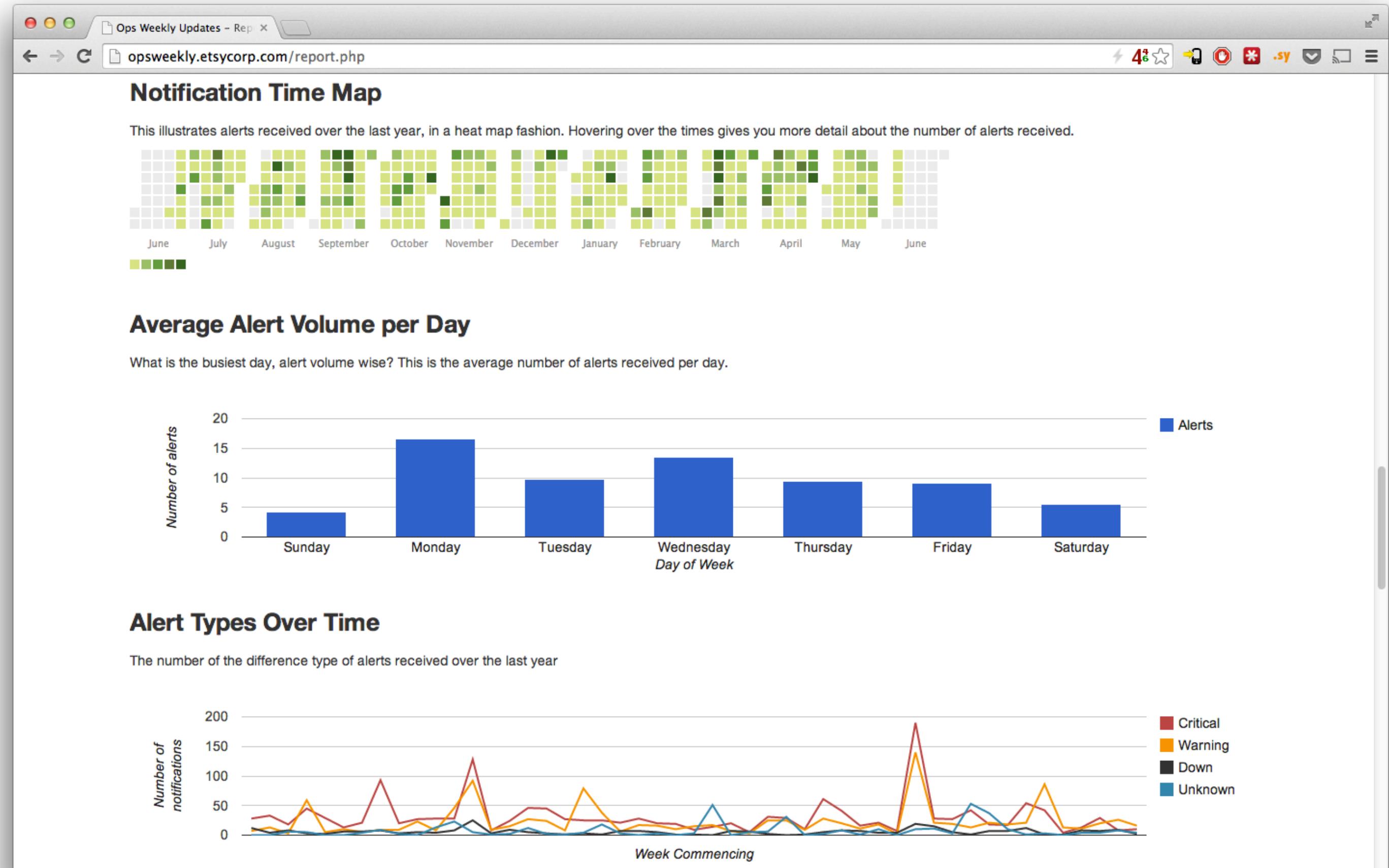
Tag Status Summary

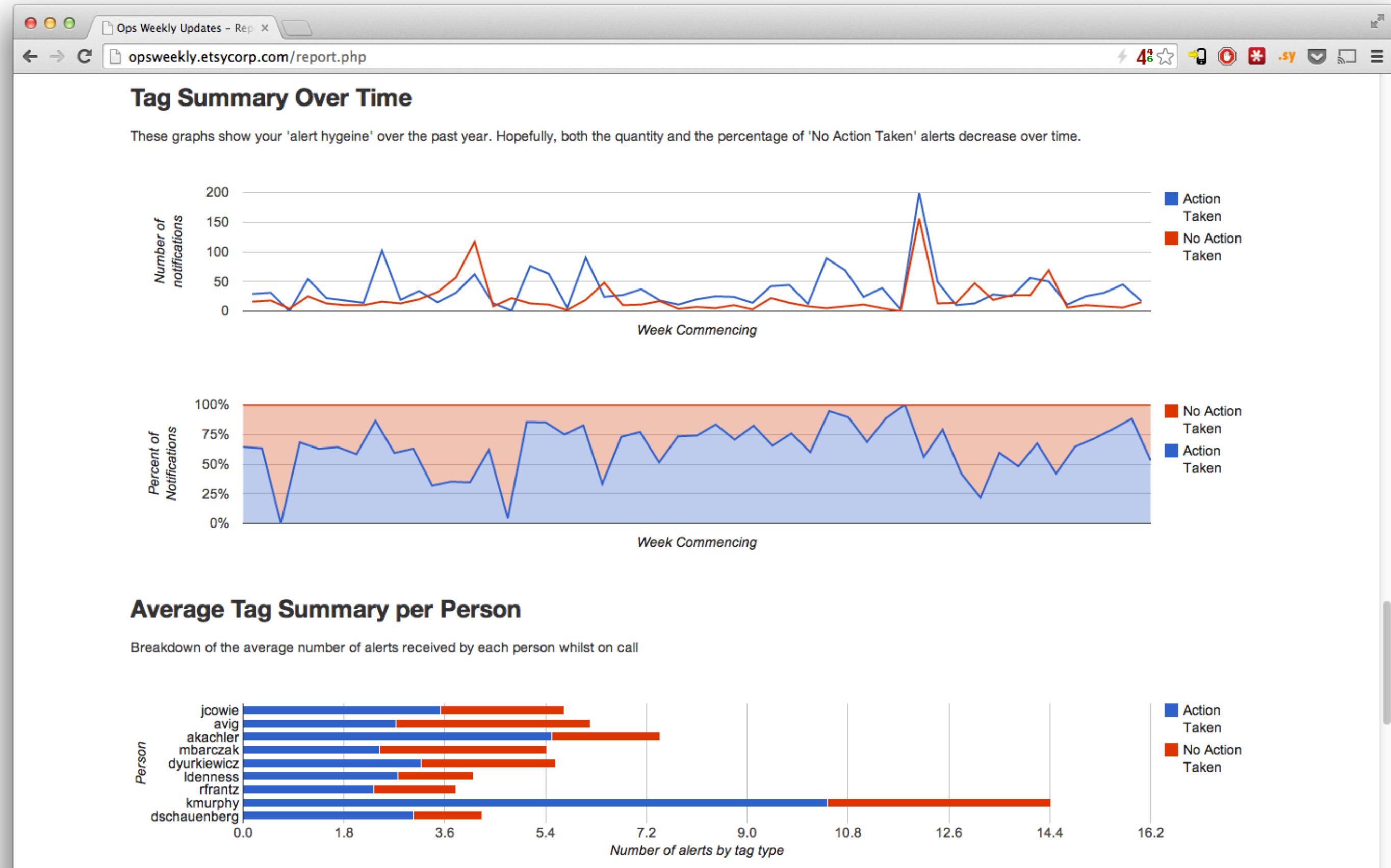
Breakdown of the tags applied to the notifications received during the week

No action taken: Check is faulty/requires modification	3 (5%)
N/A	20 (33.33%)
Action taken: Service Issue (View clean)	31 (51.67%)
No action taken: Work ongoing, downtime not set	5 (8.33%)
Untagged	1 (1.67%)

Breakdown of the tags applied (normalised)

No Action Taken	8 (13.33%)
Untagged	21 (35%)





Impact on Quality

- Service quality (SLAs)
- Visibility of quality

Statsd

```
>>> import statsd  
>>>  
>>> timer = statsd.Timer('MyApplication')  
>>>  
>>> timer.start()  
>>> # do something here  
>>> timer.stop('SomeTimer')
```

```
>>> import statsd  
>>>  
>>> counter = statsd.Counter('MyApplication')  
>>> # do something here  
>>> counter += 1
```

```
>>> import statsd  
>>>  
>>> average = statsd.Average('MyApplication', connection)  
>>> # do something here  
>>> average.send('SomeName', 'somekey:%d'.format(value))
```

Graphite

value of change

Value of Availability

- Better for customers
- Better for employees (internal services)
- Fewer pages

Value of Quality

- Deploys take less time
- Also better for customers
- More visibility into issues

Retrospective



Review

- Recognizing your Devops Narrative
- Application Deployment Planning
- Infrastructure as code
- Introducing repeatable, testable change
- Measuring impact and value of change

Next Steps

- Manual, Automation to Continuous "X"
- Be the storylistener and storyteller in your org
- Effective Devops



thank you! ❤

@sigje

