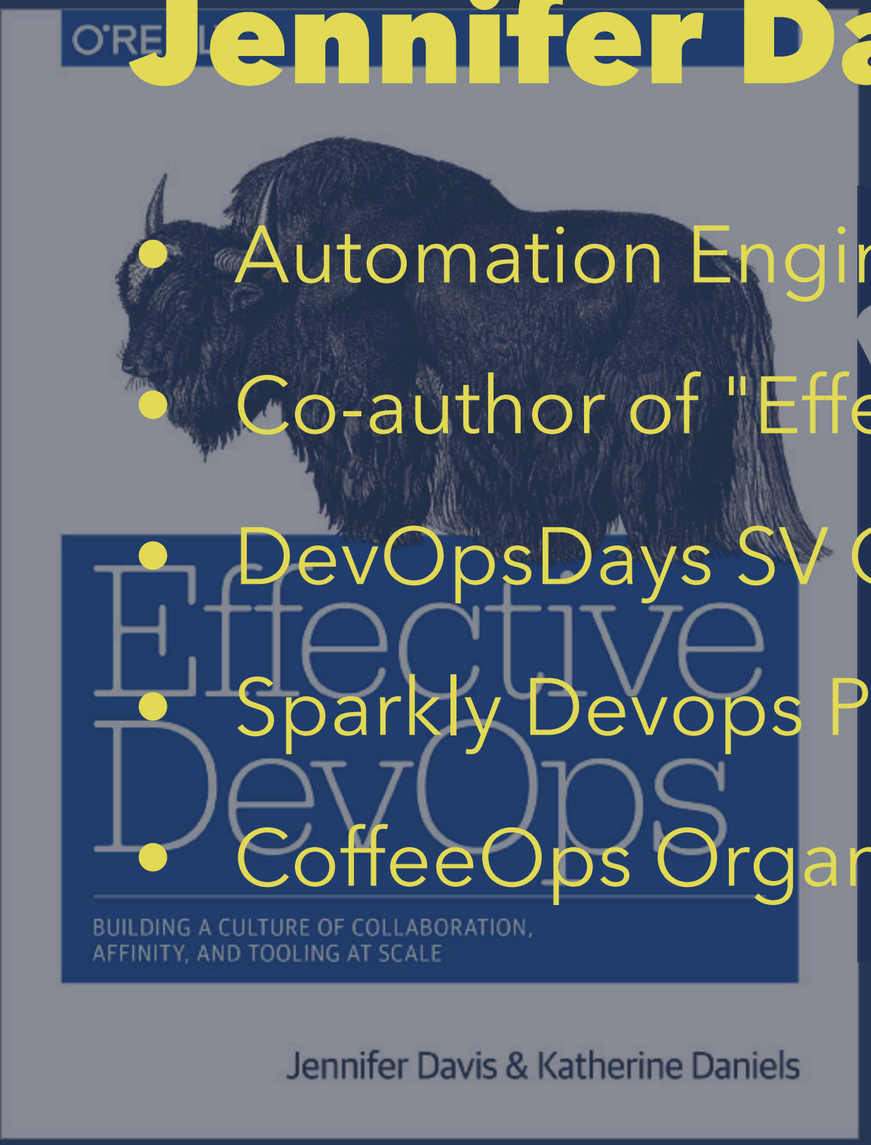
Effective Devops: Collaboration and Tools

Jennifer Davis & Katherine Daniels

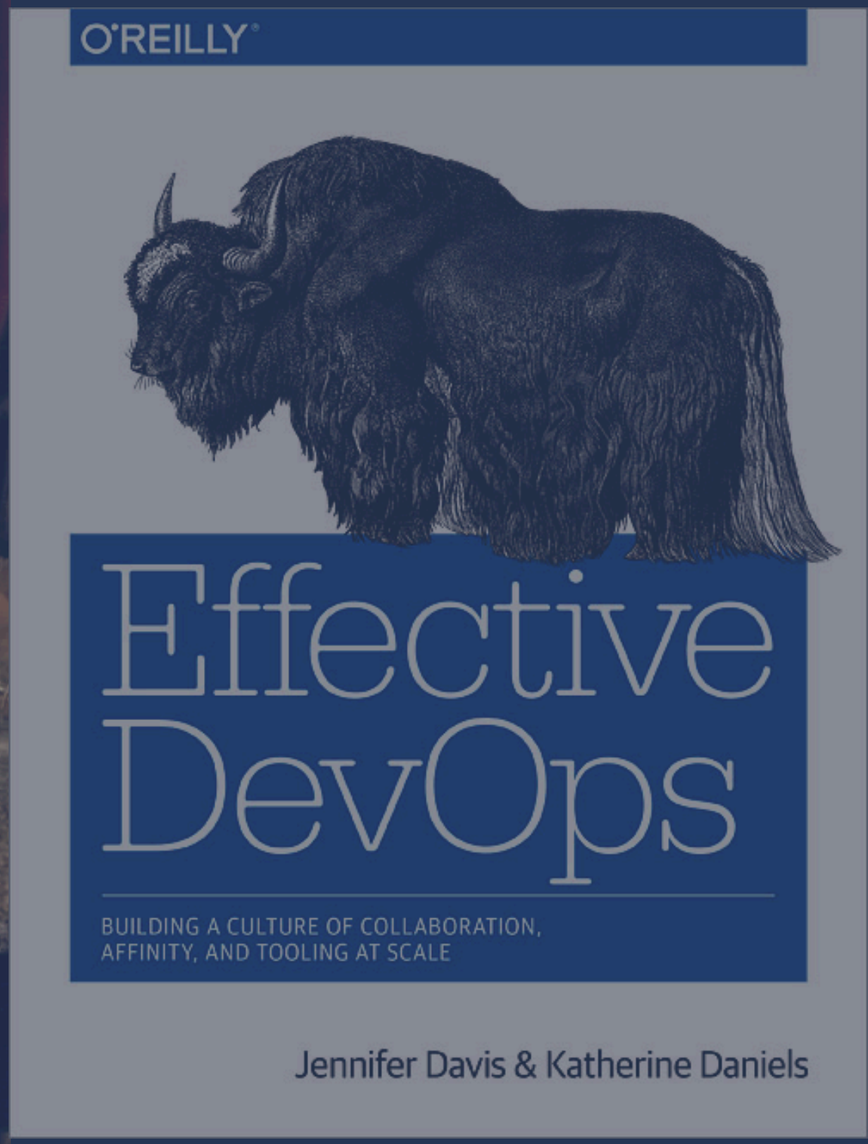
Jennifer Davis



- Automation Engineer, Chef
- Co-author of "Effective Devops"
- DevOpsDays SV Organizer
- Sparkly Devops Princess
- CoffeeOps Organizer

Katherine Daniels

- Web Operations Engineer, Etsy
- Co-author of "Effective Devops"
- DevopsDays NYC Organizer
- Ladies Who Linux NYC Organizer
- Ship Show Podcast Co-host



Communication

- Jennifer Davis
  Twitter: @sigje
  Email: iennae@gmail.com
- Katherine Daniels
  Twitter: @beerops
  Email: sparklyyakshaver@gmail.com

#EffectiveDevops

Feedback

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

Schedule

- Introduction to teams, devops principles
- 10:30-11am Morning Break
- Visualization of work, Git, Infrastructure automation
- 12:30-1:30pm Lunch

Schedule

- Testing infrastucture automation and other changes
- 3-3:30pm Afternoon Break
- Measuring, monitoring, and wrap-up

Network Connectivity

Network Name: Conference_Private

Access Code: velocity2015

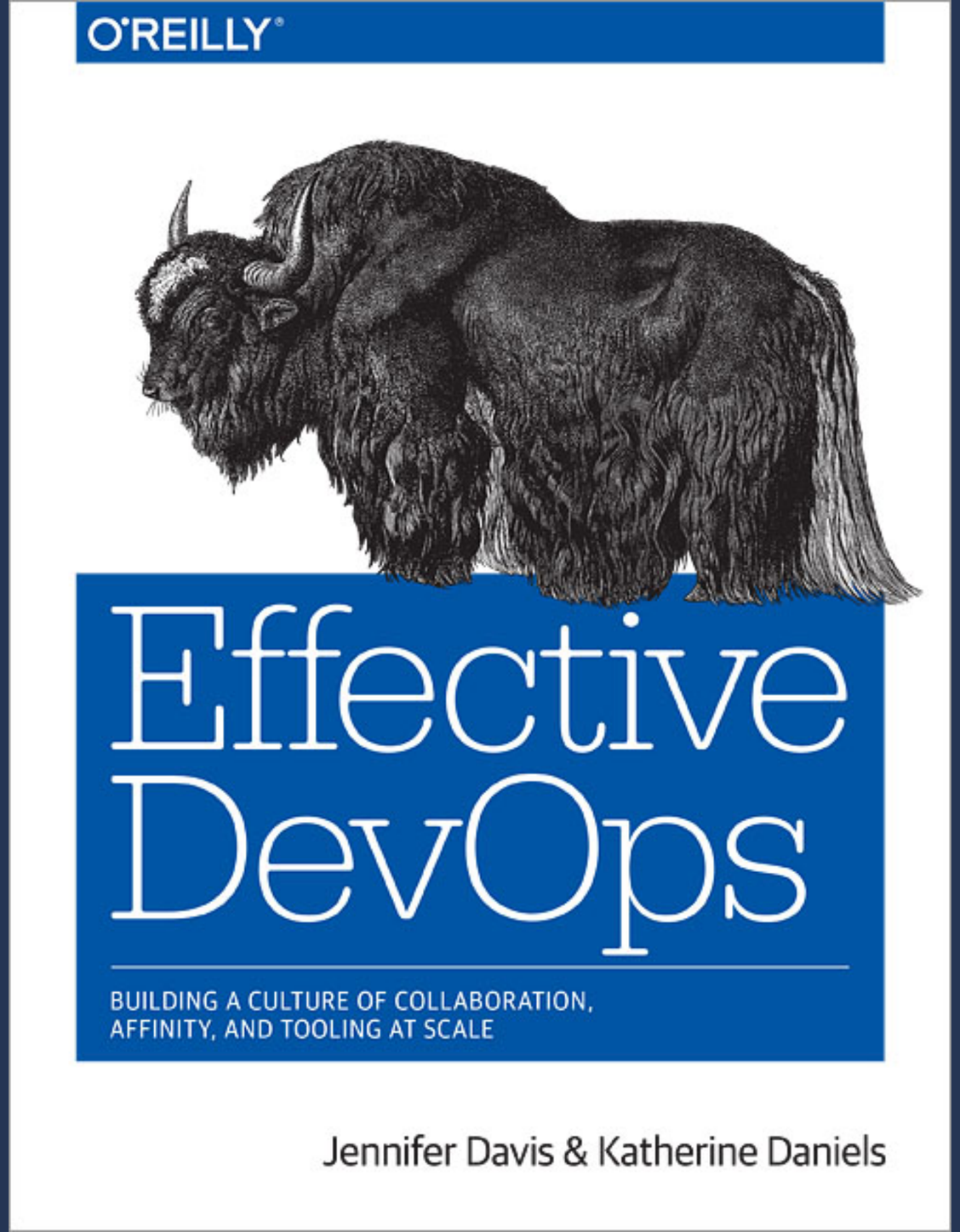
Expectations

- 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 Introductions

- Meet your team!
- Identify your team's...
  - Speaker
  - Gatekeeper
  - Notetaker

Time: 10 minutes


O'REILLY
Effective DevOps
BUILDING A CULTURE OF COLLABORATION, AFFINITY, AND TOOLING AT SCALE
Jennifer Davis & Katherine Daniels

What is Devops

- Technical cultural weave that shapes how we work, and why

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

Five Pillars of Effective Devops

- Collaboration
- Hiring
- Affinity
- Tools
- Scaling

Collaboration

- Individuals Working Together

Hiring

- Choosing Individuals

Affinity

- From Individuals to Teams

Tools

- Choosing and Using them

Scaling

- Growing and Decreasing

Collaboration and Tools

Recognizing your Devops Narrative

The Devops Compact

- shared mutual understanding
- established boundaries

Stormtrooper Syndrome

- Agency
- Adaptability: Role adherence

Learned Helplessness

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

-- George Howard

Swift Trust

- Skillful self disclosure
  - What you share, how you share it
  - Collective tasks vs personal agendas

Team

- Common purpose
- Defined beliefs
- Empowered

Small vs Large teams

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

Cultivating Empathy

- Collect stories
- Listen
- Circle back

Smarter Teams Build Better Value[1]

- Lots of Communication
- Contribute equally to team's discussions
- Theory of Mind
- Increased diversity

[1] https://github.com/thieman/github-selfies

Critical Habits for Teams

- Code Review
- Pairing

Code Review

- Max 90 minutes in one setting

Pairing

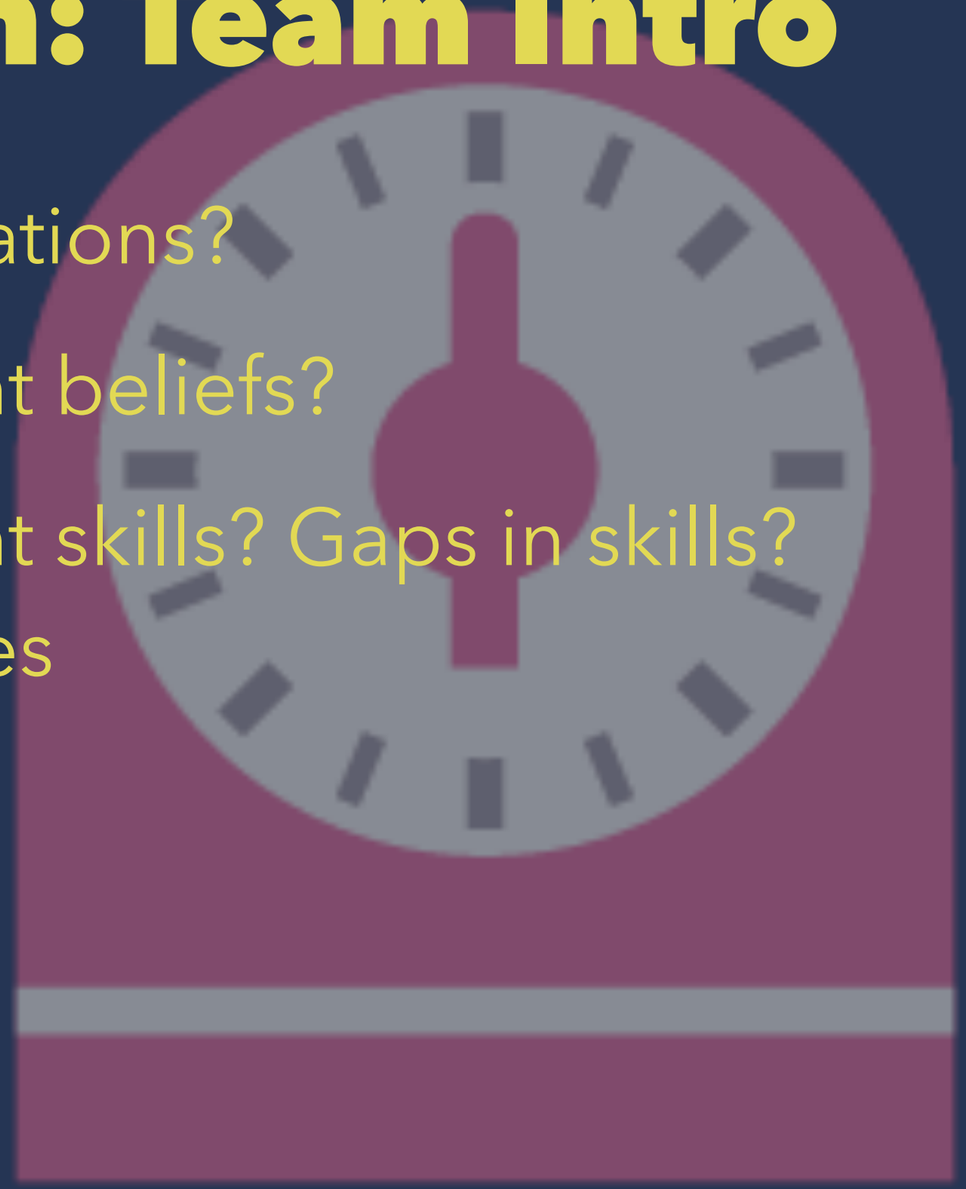
- Agile software development
- 2 people work together on 1 workstation
- Driver - writes code
- Observer - reviews each line
- Roles switch frequently

Types of Pairing

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

Discussion: Team Intro

- What are motivations?
- What are current beliefs?
- What are current skills? Gaps in skills?
  Time: 15 minutes



Share: Team Intro

- Team Name:
- Common Motivations?
- Skills? Gaps?

Fluxx

- Game about change.
- Rules change as you play.
- Goal changes frequently.

Fluxx

- Starts out with 1 Rule - Draw 1, Play 1
- Find this card (white backing)
- Place card in the middle of table

Fluxx

- Shuffle deck
- Deal 3 cards to each player

Fluxx

- First player is the person to left of Dealer.
- Start out following the Draw 1, Play 1 rule until overridden by new rules.

Fluxx

4 types of cards

- Goal - pink
- Keeper - green played in front of you
- Action - blue used once and discarded
- Rule - yellow

Fluxx - On your turn

- Draw the number of cards currently required.
- Play the number of cards currently required.
- Discard down to hand limit (if any hand limit).
- Discard down to keeper limit (if any keeper limit).

Fluxx

- Winner is the first to meet the current Goal condition.

Time: 15 minutes

Retrospective

Change is inevitable.

Importance of Games

- Build resilience
- Build collaboration (or competition)
- Roles change

Specific to Fluxx

- Frustration of constant changing goals
- Completed "work", lost value
- Visualization of goals, everyone in alignment

Visualization of Work

- Bug/issue queue
- Kanban

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

Intro JoeNGo

Application Deployment Planning

Deming Cycle

- Plan: Identify and Analyze problem
- Do: Develop and Test potential solution
- Check: Measuring effectiveness
- Act: Implement solution

Application Deployment Planning

- (L)inux
- (A)pache
- (M)ySQL
- (P)HP, Perl, or Python

Group Discussion

Time: 15 minutes

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

Nodes

Countoff

http://bit.ly/1Fg9rxs

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

Introduction to Git and Workflows

Isolated Development Environments

- Not automatically updated
- Manually pull upstream commits

Workspace Environment

```
$ git init
$ git add .
```

adds all files and directories to version control

Commit

```
$ git commit -m "message about commit" -a
```

All these changes are local! How do you collaborate?

Show all remotes

```
$ git remote -v
```

Add remote server

```
$ git remote add NAME URL
```

Update changes to remote directory

```
$ git push REPONAME BRANCH
```

View the branches

```
$ git branch -a
```

Creating the branches

```
$ git checkout -b BRANCHNAME
```

Merging branches

```
$ git merge BRANCH
```

git fetch

- import commits, from remote to local

```
git checkout master
git log origin/master
git merge origin/master
```

Different workflows

- feature branch
- gitflow
- forking
- centralized

Pull Requests

- collaboration prior to integration
  - discussion
  - follow up commits
  - selfie gifs[1]

[1] https://github.com/thieman/github-selfies

git pull

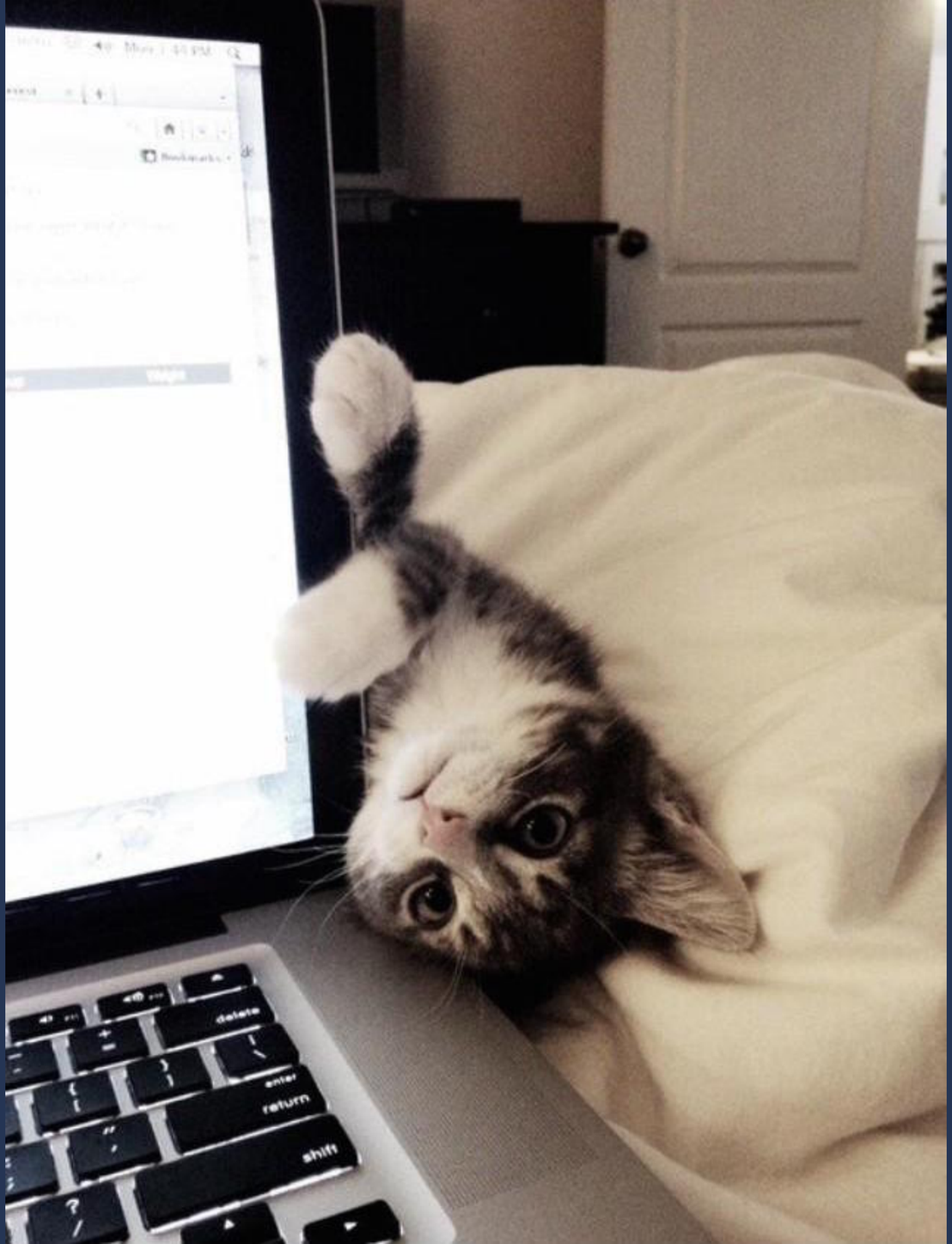
git pull REMOTE

git pull --rebase

- ensure linear history by preventing unnecessary merge commits

git push remote branch

- transfer commits from a local repo to a remote repo.
- counterpart to git fetch


delete
enter
return
shift

Assignment 1

Effective Devops

http://bit.ly/1JVuLh4

Time: 15 minutes

Getty Images

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
- CFEngine

Introduction to Chef

Resources

- Ingredients of infrastructure
- Basic building blocks

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

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

Docker

- Images
- Registries
- Containers

Assignment 2

Time: 20 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

Rubocop

- Ruby linter
- Ruby style guide
- Included with ChefDK

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

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
```

Assignment 3

Time: 10 minutes

Retrospective on Assignment 3

Time: 15 minutes

Road Runner - High Speed Access
>> Click Here! <<

Assignment 4

Time: 20 minutes

TANDY 1000

Automated Test and Build

Test, Monitor, or Diagnostic[2]

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.

Measuring Impact and Value of Change

Impact of Change

Impact on Availability

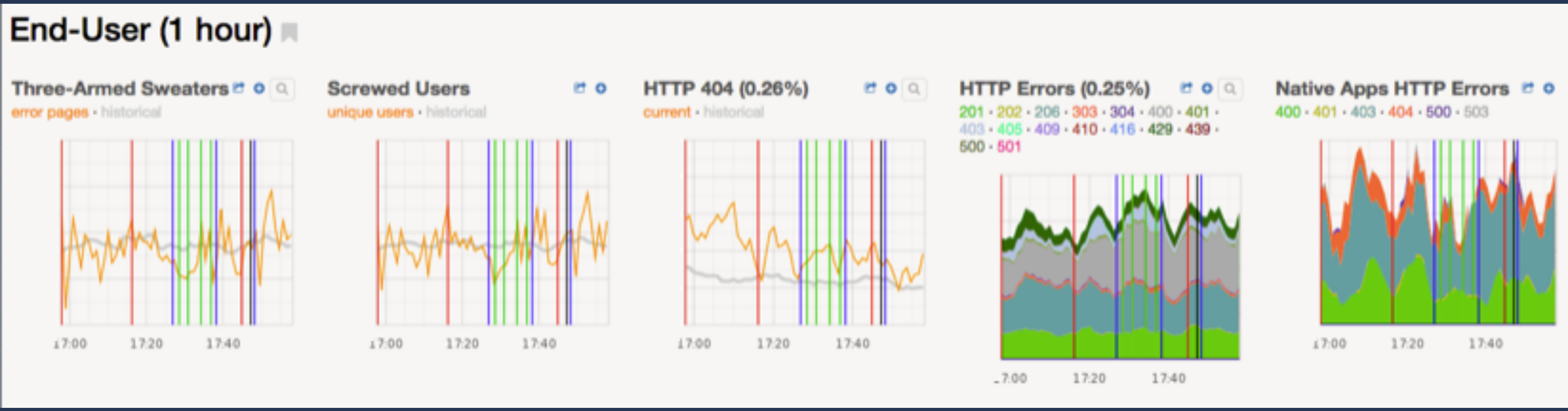
- Overall site/app availability
- Individual service availability

Availability Monitoring

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

End-User (1 hour)

Eventinator


End-User (1 hour)
Three-Armed Sweaters
error pages · historical
17:00
17:20
17:40
Screwed Users
unique users · historical
17:00
17:20
17:40
HTTP 404 (0.26%)
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

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

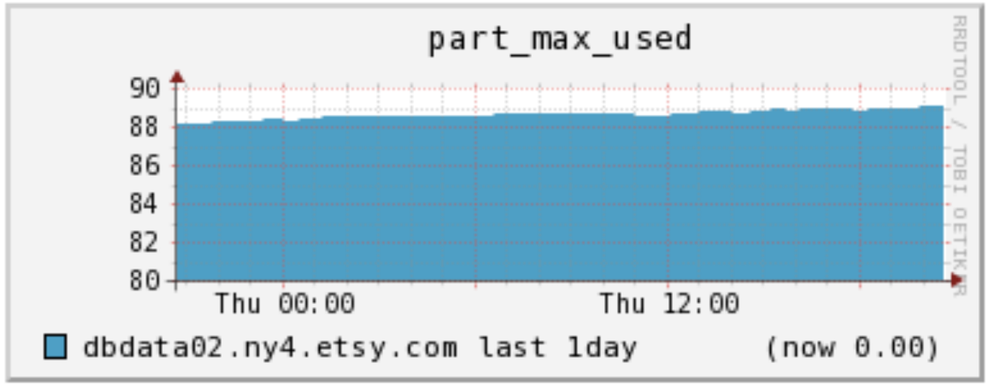
to me

Host: dbdata02.ny4 **Service**: Disk Space

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

/var

Problematic volume



Ganglia graph

```
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
```

Threshold exceeded

Alert Frequency

Alert frequency

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

OpsWeekly

Ops Weekly Updates - Add

opsweekly.etsycorp.com/add.php

Opsweekly | Overview | Add | Reports | Meeting | Search Reports and Notifications | Set Date | Timezone | Idenness

Update for week ending Sunday 15th June 2014

On call report

Alerts recieved this week (Friday 6th June 2014 - Friday 13th June 2014)

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	
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 clea		Notes:	RAID failure caused machine to die	
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 clea		Notes:	MySQL slave failure on host	
Thu 12 Jun 19:12:43 EDT	database001b	Host Check	(Host Check Timed Out)	DOWN
Action taken: Service Issue (View clea		Notes:	Machine died, hardware failure	
Wed 11 Jun 12:57:21 EDT	api05	Memory	CHECK_NRPE: Socket timeout after 30 seconds.	UNKNOWN

Ops Weekly Updates - Rep

opsweekly.etsycorp.com/report.php

Opsweekly | Overview | Add | Reports | Meeting | Search Reports and Notifications | Set Date | Timezone | Idenness

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 recieved 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 recieved 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)

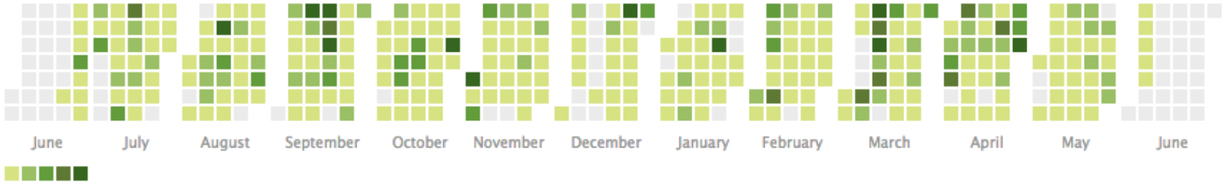
No Action Taken	8 (13.33%)
Untagged	21 (35%)

Ops Weekly Updates - Rep

opsweekly.etsycorp.com/report.php

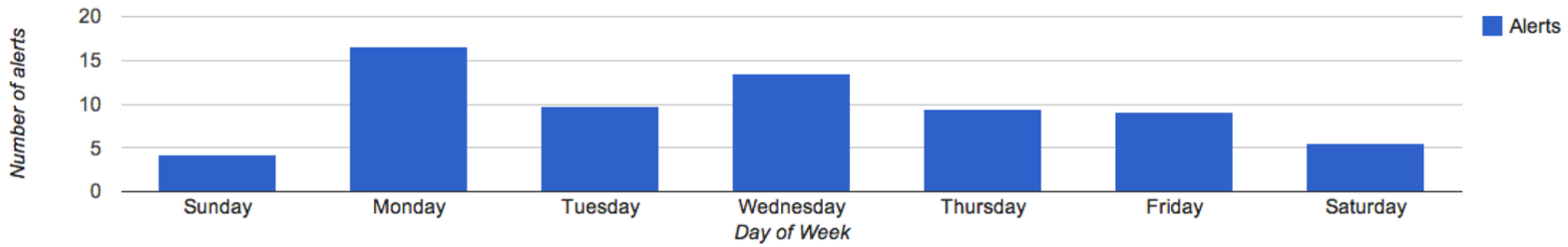
Notification Time Map

This illustrates alerts received over the last year, in a heat map fashion. Hovering over the times gives you more detail about the number of alerts received.



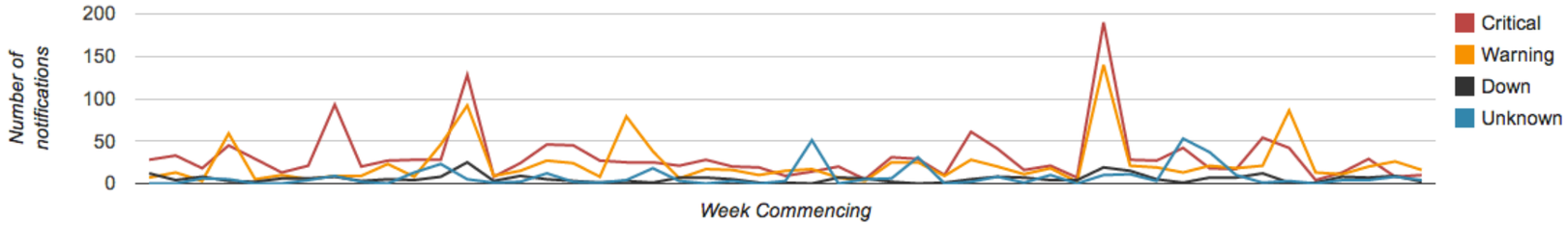
Average Alert Volume per Day

What is the busiest day, alert volume wise? This is the average number of alerts received per day.



Alert Types Over Time

The number of the difference type of alerts received over the last year

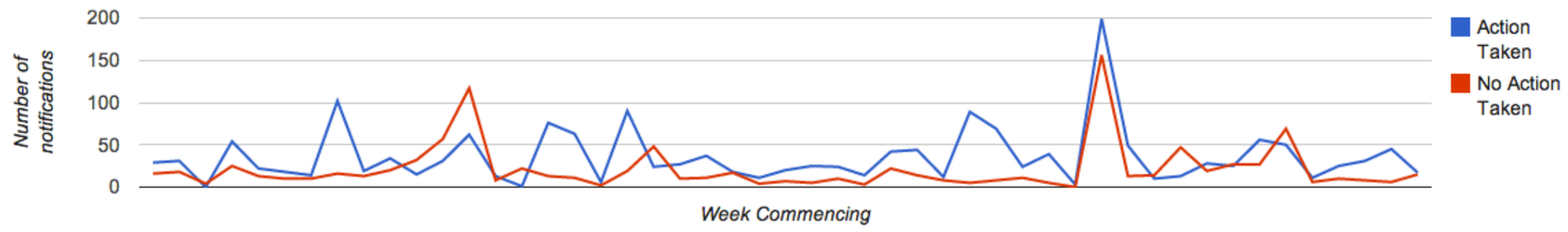


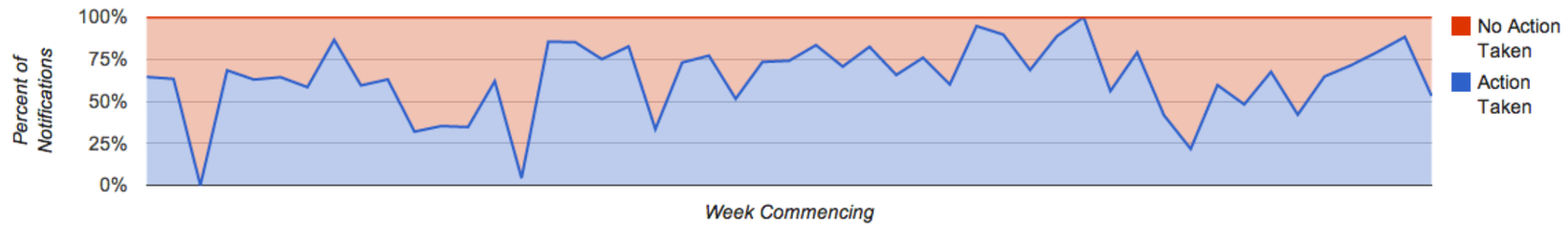
Ops Weekly Updates - Rep

opsweekly.etsycorp.com/report.php

Tag Summary Over Time

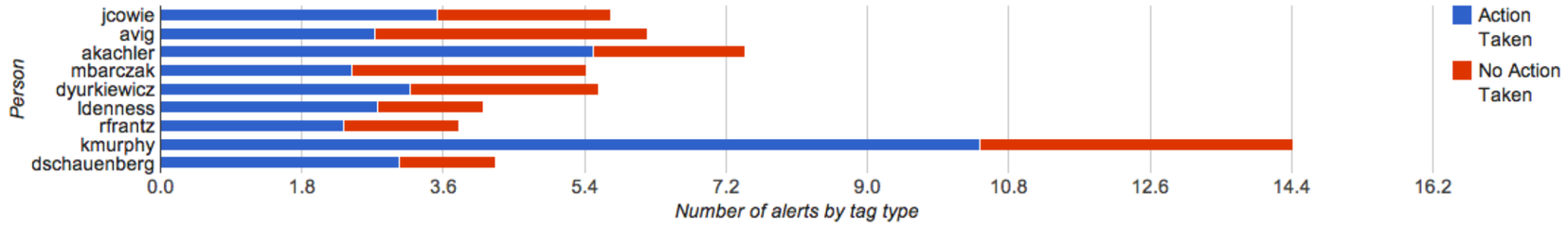
These graphs show your 'alert hygeine' over the past year. Hopefully, both the quantity and the percentage of 'No Action Taken' alerts decrease over time.





Average Tag Summary per Person

Breakdown of the average number of alerts received by each person whilst on call



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

Assignment 5

Time: 20 minutes

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 available in Early Release

Slides

Thank you! ❤

@sigje

@beerops

Further Resources

- Chef and Containers
- Serverspec