PAGERDUTY

Arup Chakrabarti

OPERATIONS ENGINEERING

arup@pagerduty.com

@arupchak

Common Ops Mistakes and How to Prevent Them

PAGERDUTY

Quick Bio

Who the heck is this guy?

- Worked at Amazon as an Engineer/Manager
- Worked at Netflix as a Manager
- Employee #20-something at PagerDuty
  - Infrastructure was a monolithic Rails app and a single service
  - Still have the MonoRail, now with 10+ services
  - Over last year, ~20 servers -> ~200 servers

Quick Disclaimer

- I did not come up with everything
- I am speaking for myself

What is Software Operations?

The Magical Formula

What is Software Operations?

The Magical Formula

- Change ~ Downtime
- More change => More Problems

Why this is scary

Let's Revise the Magical Formula

Why this is scary

Let's Revise the Magical Formula

- Changes ~ Innovation ~ Downtime
- Maintain stability by stopping innovation
- Scrappy Startup vs. Big Company
- Most Big Companies do not innovate because they cannot risk the change
- Does this mean all companies are eventually doomed to not innovate?

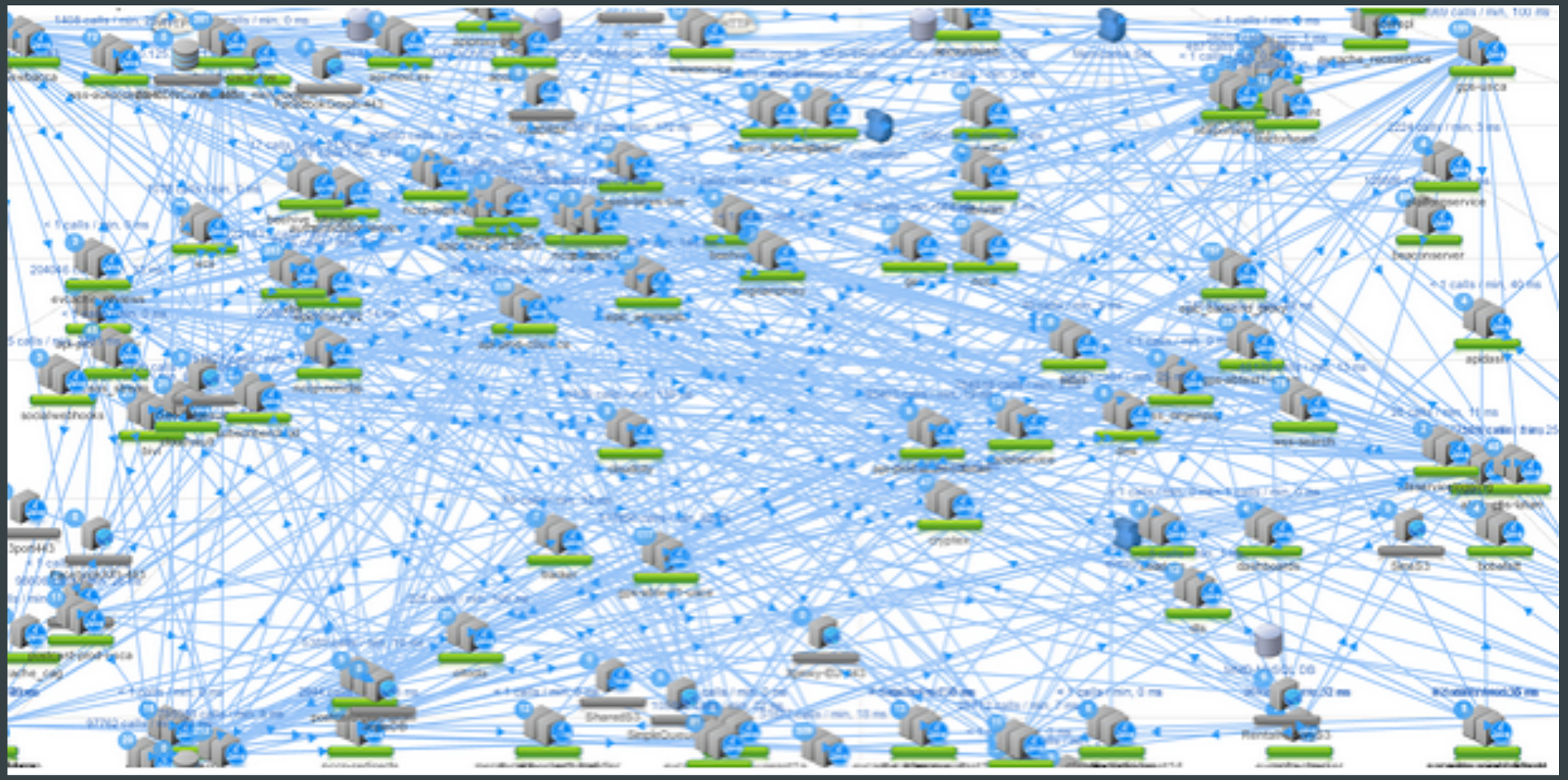
What is Software Operations?

The Magical Formula Revised Again

- Changes ~ **(k)** Innovation ~ **(h)** Downtime
  - There are two constants - **k** and **h**
- **k** - Increase **k** to amplify innovation per change
  - Test environments, A/B testing, splitting up codebases
- **h** - Decrease **h** to improve stability per change
  - Fast deploys, better alerting, splitting up codebases

0. Accept that no infrastructure is perfect

Netflix Architecture Diagram



0. Accept that no infrastructure is perfect

Really, it is ok

- Make the best decisions at the time
- Accept constraints
- Learn more as our systems or business evolve
- This is ok

1. Initial Setup of Infrastructure

- Using personal email accounts
  - No, setup mailing lists, ideally have Google Apps setup from the beginning
- Pre-Optimizing for Scale
  - Use Heroku or other PaaS for as long as you can
- Technology selection
  - Boring technologies to do cool things
- Password storage
  - Not in the git repo, use ENV vars or your configuration management tool

2. Proper Test Environments

- Separate hosting account for testing
- Have separate provider accounts for test (e.g. email providers)
- For local development, use VMs
- Do no run services on localhost
- Use Vagrant for this

3. Configuration Management

- Early on, use Ansible or Salt
  - Light weight and easy to learn
  - Enforces treating ‘Infrastructure as Code’
  - Will scale just fine when you only have 4-5 server types
  - Avoid Bash Scripts
- Beyond 5 server types, move to Chef, Puppet, Asgard, or other heavier tools
- Augment Cloud Formation or other PaaS specific tools

4. Deployments

- Consistency
  - Every Engineer
  - Every Piece of Code
- Use some orchestration tool with Git
  - Capistrano
  - Ansible
  - Salt
  - Celery

5. Incident Management

- Have a process in place and document somewhere that is easily shared
  - Wiki
  - Dropbox document
  - Not in a random email
- Make sure you review it monthly

5. Incident Management

Example Procedure

1. Everyone jumps onto chat client
2. Everyone dials into group call
3. Each member of the team gives a status update
4. Single person acts as call leader (not a resolver)
5. Call leader gives out orders
6. Have a status update every 10 minutes
7. Call leader maintains an outage log
8. Conduct a post-mortem

6. Monitoring and Alerting

- Start with anything
  - StatsD with Graphite backend
  - CloudWatch
  - Sensu
  - Nagios
- Use hosted solutions (as long as they make fiscal sense)
  - New Relic or other APM's

7. Backups

- Backup your data regularly to S3
- Test your restores at least monthly
  - Practice restoring production data to test env
  - Make sure to scrub sensitive data
  - Measure time to recovery

8. High Availability 101

- Multiple servers at every layer
- Multiple Load Balancers in DNS
- Multiple App Servers
  - App servers have to be stateless
- Use Clustered Datastores
  - MySQL XtraDB Cluster
  - Cassandra
  - Avoid Master/Slave architectures
- Worry about sharding later
  - You do not know what to shard on yet

9. Security 101

- Use Gateway Hosts for SSH
  - These hosts are whitelisted for SSH, everything else should have global SSH turned off
- Unique user accounts for everything
  - Easy to revoke access when something happens
- Use PaaS security features
  - Security Groups, VPC, etc
- SSL encryption on everything

10. Internal IT needs

- Have a central list of tools that every department needs
  - Onboarding docs are a good place for this
- Consolidate machine types
  - Do not let everyone have every machine that they want
  - Easier to support and swap out machines
- Use images for machines
  - Easy to take a USB stick and make a general image

Exploiting your business patterns

for managing change

- Look for seasonality in traffic patterns
  - You can make changes when traffic is at the trough
- Look for where you can be latency tolerant
  - Can you tolerate an extra 100-200ms of latency?
- What gets impacted when you go down?
  - Actual revenue
  - Customer trust

Thank you.

Slides will be available at https://speakerdeck.com/arupchak

Arup Chakrabarti

OPERATIONS ENGINEERING

arup@pagerduty.com

@arupchak

PAGERDUTY