Strategies for adopting TDD in Operations

@RanjibDey
Agenda

1. Introduction
2. TDD in brief
3. Why TDD is compelling in Operations
4. Strategies
5. Key learnings
6. Q/A
Introduction
TDD in brief

1. Unit test first
2. Part of XP
3. Has evolved a lot
Why TDD in Operations?

Operations integrates services
Why TDD in Operations?

Tactical workloads are difficult to predict
Why TDD in Operations?

Legacy components
Strategies

1. Layered tests
2. Code reviews
3. Extend CI to operations
4. Continuous learning

@RanjibDey
Layered testing strategy

1. Unit tests for fast feedback
2. Functional tests for feature validation
3. Integration tests for end to end testing
Unit tests

directory "/var/go/.ssh" do
  owner  'go'
  group  'go'
  mode   0700
end

it 'creates the go user ssh directory' do
  expect(runner).to create_directory('/var/go/.ssh').with(
    owner: 'go',
    group: 'go',
    mode: 0700
  )
end
Unit tests

$ rake spec
Finished in 1 minute 15.16 seconds
5629 examples, 0 failures, 4 pending
Functional tests

it 'creates a standalone zk node' do
  ct = helper.container('pd-zk')
  out = ct.execute do
    h = 'localhost'
    p = 2181
    telnet = Net::Telnet::new('Host' => h, 'Port' => p)
    state = telnet.cmd('String' => 'ruok')
    telnet.close
  state
  end
  expect(out).to eq('imok')
end
Integration tests

@RanjibDey
Code reviews

(chore)add jruby in travis build #48

@RanjibDey
Continuous Integration

Extend CI in operations
Continuous Integration

Make courtesy artifacts

@RanjibDey
Continuous Learning

1. Invest on learning resources
2. Conferences
3. Engage in OpenSource projects
Continuous Learning

Invest on learning resources

@RanjibDey
Continuous Learning

Conferences, meetups

@RanjibDey
Continuous Learning

Contribute and learn from OpenSource projects

- **sethargo / chefspec**
  - Write RSpec examples and generate coverage reports for Chef recipes! [http://sethargo.github.io/chefspect](http://sethargo.github.io/chefspect)

- **opscode-cookbooks / dynect**
  - Development repository for Opscode Cookbook dynect [http://community.opscode.com/cookbooks/dynect](http://community.opscode.com/cookbooks/dynect)

- **lxc / ruby-lxc**
  - ruby bindings for liblxc [https://linuxcontainers.org](https://linuxcontainers.org)

@RanjibDey
Key Learnings

1. Be patient
2. Capture patterns, test patterns
3. Embrace diversity
Key Learnings

Be patient
Key Learnings

Capture patterns

@RanjibDey
Key Learnings

Capture patterns

Infrastructre Tooling Patterns List

The DevOps Zone is brought to you in partnership with New Relic. Improving the performance of your app is easy with New Relic’s SaaS-based monitoring.

Following is my grouping of tools that I have learned/used as a sysadmin and DevOps dude at ThoughtWorks while maintaining our distributed infrastructure, setting up our private cloud installations, and in many different client gigs.

You can add some of these tools when required as your infrastructure/deployment/app grows.

1. **Provisioner**: Abstracts your vm / environment provisioning mechanism. Mostly relevant if you are on cloud infrastructure. Examples are boto and fog. Very important if you plan to do something like auto scaling. Gives you elastic infrastructure.

2. **Configuration Management system**: Lets you create reusable environments by expressing packages, services, files and other components via a DSL. It also addresses cross platform issues. Puppet, Chef, Cfengine, and Salt are examples. A mature CMS setup will give you context aware infrastructure, like how your web server can automatically recognize the DB server, or how the load balancer can automatically recognize your web servers. A mature CMS setup will also incorporate the notion of environments and have versioned infrastructures like UAT, which can have app deployment version 1.3 and production version in 1.1 and staging in 2.0 etc.
Key Learnings

Embrace diversity, ease deduction

@RanjibDey
Thank You