# Devops Loop Tutorial

Systems Administrator Focused
Who, What, When, Where, Why

# Who am I?

Richard Clark
richard@craftypenguins.net

- General Public?
  - I do computer stuff
  - I clean up messes
- Inquisitive / Technical?
  - I clean up messes
  - Systems/Devops architecture
  - Software Engineering
  - Embedded Systems architecture
  - CKA - I am biased to Kubernetes
  - UI/UX Design
  - Sales Engineer

Crafty Penguins - craftypenguins.net

- Kerkhoff Technologies Inc.
- Everything Linux-ish
- SaaS support services
- Devops and Automation
- Software Engineering
- Repair, Manage, Create

# Goal

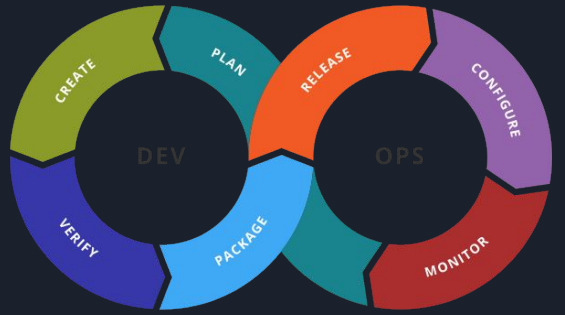
“We have had a lot of presentations from Devops vendors on their systems and tools, but no one has shown us how one actually goes about doing it.”

- Incose Rep (paraphrased)

# Agenda

A. Quick run through of what devops means to us at Crafty Penguins
B. Introduction to the “product” we will take through “being devops’ed”
C. The Twelve Factor App
D. Version Control System.  Git.
E. Testing - “Unit” testing
F. Continuous Integration (CI)
G. Containerization
H. Testing - “Integration” Testing
I. Continuous Delivery (CD)
J. Metrics, Monitoring, Alerting

# What devops means to us?



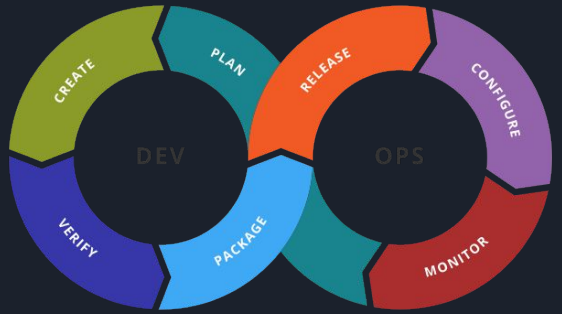

- Devops: Dev to Ops and back again
- Developers (Issues)
  - Having to create and test code locally instead of production
  - Iterate faster to production - Get that feature out!
- Systems Engineer (Issues)
  - Having to juggle lots of (Applications x Audiences) with devs pushed to “get that feature out”
  - Application count going up with Microservices

# What “doing devops” means to us?

- Depends on who is telling the story
- We’re Devops Engineers
  - Mix of software and systems engineering skills
  - Some of us focus more on Development side devops
  - Some of us focus more on Systems Infrastructure side devops
- Team effort
  - Changes in “the product”
  - Changes in the networking
  - Changes in the hardware
  - Devops goes through Devops

“Doing Devops”

Creating and configuring the “product”, the tooling, and the systems infrastructure (networking, metal, monitoring) that keeps the above loop running smooth.



# Our Demo Product

Devops vendors are supplying the tools that

- Raise the speed
- Lower the skill

You still have to get the materials and build the **house...**

## Small devops run through the forest.  A ‘bird house’

- Front End SPA (Javascript)
- Back End API (TBD)
- Database (TBD)
- We will go through, at a simple level, the pieces we need to put together for the “devops” flow of our product from dev to production

# “AppName”

Simple Grocery List Application

- Quasar Framework (Vue.js) single page application (SPA)
- Will hit an API backend to store and load list items
- API Backend is written in TBD
- API will fetch data from a database (TBD)

{{ TODO Run quick demo of app / insert animation background here}}

# 12 Factor App

https://12factor.net/

1. **Codebase - One codebase tracked in revision control, many deploys**
2. **Dependencies** - Explicitly declare and isolate dependencies
3. **Config - Store config in the environment**
4. Backing services - Treat backing services as attached resources
5. **Build, release, run - Strictly separate build and run stages**
6. **Processes - Execute the app as one or more stateless processes**
7. **Port binding** - Export services via port binding
8. **Concurrency** - Scale out via the process model
9. **Disposability - Maximize robustness with fast startup and graceful shutdown**
10. **Dev/prod parity - Keep development, staging, and production as similar as possible**
11. **Logs** - Treat logs as event streams
12. **Admin processes** - Run admin/management tasks as one-off processes

# Version Control - Git, just git

1 - One codebase tracked in revision control, many deploys

Not all cars have 4 wheels, round steering wheels, and a brake pedal.

Not every revision control system needs to be GIT, but….

# Version Control

1 - One codebase tracked in revision control, many deploys

Why do we need revision control?

- Programmer - Undo / Experiments
- Team - Merging everyone's work
- **Devops - We need to detect and process change sets**

# DEMO - VCS System

Gitlab / Our Code Repositories

# Questions?

On GIT, Software Revision control

# Testing - “Unit” testing

- Created along side the code by Developers / QA Team
- Usually tests code sections input and output
- Test Driven Development
- External components are “Mocked” out
- Why do we care for Devops?
  - First layer to reject code before doing larger longer tests.  Catch regressions.
  - We will be testing code in various environments - Does it still pass it’s basic test
  - We will be running the code through various conditions
- First thing to get the axe under deadline pressure.
  - Airline regulations are written in blood

# DEMO

Add some unit testing

Run a test

Fix the code

Run a test

Test “Output”

# Questions?

On unit testing, mocking

# Continuous Integration (CI)

Developers make isolated changes

- Works for me

CI process will validate that the merged version works

- Takes a proposed change
- Merges it with a “main line” branch (master, prod, develop)
- Runs a suite of tests to validate the merge would not break
- Merges in the work or reports the failure to be fixed

# DEMO

Discuss the tool (Gitlab -> CI)

Add a CI command file

Make a code change

Commit the code

Make a merge request

See the failure

Fix the issue, recommit, see CI run, Close the passed merge request

# Questions?

On continuous integration / testing

# Containerization

9 - Disposability - Maximize robustness with fast startup and graceful shutdown
10 - Dev/prod parity - Keep development, staging, and production as similar as possible

- The “Product” is in version control
- ..but so far the Developer and CI environment are unique
- Works for me - Test works for Dev, but fails in CI
- We need a reference “environment”
- We want to use this everywhere that the product/tests run
- Quick to start/kill
- Security - Only add what is needed
- **Application runs as PID 1**
    - A not often realistic goal is that is the only thing there

# Questions?

# Containerization

9 - Disposability - Maximize robustness with fast startup and graceful shutdown
10 - Dev/prod parity - Keep development, staging, and production as similar as possible

- Containerization will help us with these goals
  - System managed configuration
  - Easy Deploy/Kill - Cattle vs. Pets
  - Track in version control
- Docker (as a container runtime format)
  - “ZIP” file for OS File System
  - https://opencontainers.org/
- Docker (as a tool/product)
  - Wrapper around Linux container primitives CGroup and Namespaces
  - Tooling for building images
  - Low level orchestration management to run and connect containers (CRI)

# Questions?

On “Containers”, “Docker”, “Kubernetes”, “Lightweight virtual machines”

# DEMO

Analyzing the components we want to containerize

Create repositories for these

Turn up, demo running

Highlight a missing component from dev provided guidelines (a ‘works for me’ situation)

Highlight how a developer would make use of these locally

# Questions?

Containerization  (deployment of containers yet to come)

# Containerization

## 3 - Config - Store config in the environment

- We have repeatable version controlled “Servers” now
- App Container 1.0.3 should be identical on developers system, CI/CD, QA, Staging, Production, etc. environments
- To allow that to be true, code must use RUNTIME configuration that has been added into the container at startup.

# DEMO

Highlight the ‘hard coded’ configuration issue

Change to become a ‘run time’ value

Demo the new runtime configuration

# Questions?

- Containerization
- Runtime Configuration

# Catch up Summary #1

- Our code is version managed
- Our “Servers” are version managed (first stages of Code as Infrastructure)
- Our version managed servers become environment specific at run time.
- The code has self contained tests that can validate it working in these containers
- Works for me problem is much easier to debug and catch early
- Developers can work on systems that will closely resemble production

# Integration Testing


This is why we need integration testing

# DEMO

Get the CI system to use the containers

Highlight a failure to integrate (Mocked Unit test passes, but fails with whole system)

Create an integration test

Collect the failure from CI system

Fix

Collect the pass from CI system

# Questions?

On demonstrated samples:

- Continuous Integration
- Testing

# Continuous Delivery (CD)

- It passed the tests in CI !
- Now what?

- Ship it !!
- Where?
  - Kubernetes - General purpose container orchestration - OS of the Cloud (Bare metal and soon on a Cloud Provider near you!)
  - AWS - ECS/
- When?  Move through various levels of acceptance
  - QA
  - Staging
  - Canary
  - Production

# DEMO - CD

Quick K8S overview

Hook up our CI/CD system to K8S

Have our integration test run inside K8S

Setup a deployment

Approve and test the deployment

# Continuous Delivery (CD)

..continued

- It's deployed - What next?
  - Soft level interrogate that first level deployment
  - Run the integration tests against it
  - Have a QA Team have at it (Yes, you still need a QA Team)
  - Any issues found by QA show a missing unit or integration test so push back
    - Don't FIX the failure, add a test to have it be caught before this stage
    - Then fix the failure, have those tests pass
  - Create more integration tests / acceptance tests
- Deploy to a Staging level and repeat
  - Should be as identical to production as possible
    - Resources / Performance
    - Data
  - Be evil to this deployment
    - Chaos Monkey
- QA on the public
  - A/B, Blue/Green, "Run Away" deployments
  - Canary Deployments

# Questions?

On Continuous Delivery

# MMA

Monitoring, Metrics, Alerting

- Devops is a feedback loop
  - MMA closes that loop
- Collect Data from each stage before promoting
  - Compare with previous runs to look for issues
  - Feedback into bugs/issues
- Monitor the health of production systems
  - If being down == someone is blocked, then it's production
- Log capture

# DEMO - Metrics

TSDB / Prometheus / Pull model

Grafana

Logs - ELK / Loki

Code Metrics Demo - Expose metrics and scrape, see data

# DEMO - Metrics

Quick demo of System Level metrics

Expose code level metrics

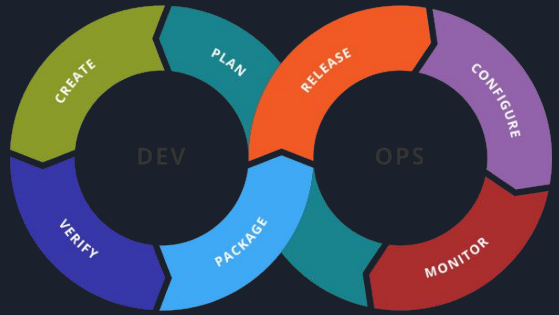
Log recording

# Questions?

On Metrics, Monitoring, and Alerting

# Look back on what we have covered

A. Quick run through of what devops means to us at CP
B. Introduction to the “product” we will take through from start to end
C. The Twelve Factor App
D. Version Control System.  Git.
E. Testing - “Unit” testing
F. Continuous Integration (CI)
G. Containerization
H. Testing - “Integration” Testing
I. Continuous Delivery (CD)
J. Metrics, Monitoring, Alerting



# What’s Next?

Everything in moderation including moderation.

- Split monoliths into sensible separate code/service block
  - Don’t get carried away - FaaS
- More Integration Testing - Thorough
- More Unit testing - Fast
- More feedback

# Questions / End

Richard Clark
richard@craftypenguins.net

- General Public?
  - I do computer stuff
  - I clean up messes
- Inquisitive / Technical?
  - I clean up messes
  - Systems/Devops architecture
  - Software Engineering
  - Embedded Systems architecture
  - CKA - I am biased to Kubernetes
  - UI/UX Design
  - Sales Engineer

Crafty Penguins - craftypenguins.net

- Kerkhoff Technologies Inc.
- Everything Linux-ish
- SaaS support services
- Devops and Automation
- Software Engineering
- Repair, Manage, Create

{{todo : Insert QR for Repo/Slides}}