DevOps workshop

Mercer

26th - 29th Mar, 2017

Introductions

- Gaurav Nagar
  - Enterprise agility catalyst
  - DevOps coach
  - Entrepreneur
  - Loves to cook!

Expectations!

- What do you think the course is about
- What do you want to learn?
- What issues do you think this workshop will help you solve?

- What will make this worth coming on Saturday?
- How do you want this workshop to be conducted?

Why are we here?

This is your last chance. After this, there is no turning back. You take the blue pill - the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill - you stay in Wonderland and I show you how deep the rabbit-hole goes.

- Morpheus (Matrix)

Workshop format

- Presentation of concepts
- Free-form discussion (FFD) around the concepts
- Hands-on exercises
- Pairing and collective discovery

Rule: Comprehension over coverage

The problem

- Team disbands after delivery – zero ownership
- Technical debts are ignored (not even measured) – Crummy code quality
- Roadblocks at each step during delivery
  - Infrastructure
  - Development frameworks and architecture
  - Security
  - Releases and deployments (manual)
  - Infrastructure parity across environments
- Results in…

Infighting!

- Us vs Them
- Not my problem
- Airspace love
- Toxic culture
- Confused business
- Trust deficit – within IT, and with business

WASTE!

- Knowledge waste
  - Outdated document, Movement of key resources, Changes become very painful
- Waiting waste
  - Waiting for servers, For testing, For deployment
- Capacity waste
  - Demand more upfront as upgrading is painful
- Motion waste
  - Work-life imbalance, Health and emotional problems
- Transportation waste
  - Environment movement
- Correction waste
  - Lots of bugs, Poor quality
- Inventory waste
  - Code does not get shipped

DevOps – A brief history

- Patrick Debois and Andrew Shafer discuss DevOps in Agile conference 2008
- Velocity 2009, John Allspaw and Paul Hammond presents "10 Deploys per Day: Dev and Ops Cooperation" at Flicker
- March 2011 – Gartner predicts explosion of DevOps adoption in Global 2000 companies in coming years

DevOps defined: Stay CALMs

- **C**ulture
- **A**utomation
- **L**ean
- **M**onitoring (or measurement)
  - Feedback cycle
  - How are we doing?
  - How soon can we recover?
- **S**haring
  - Ideas
  - Feedback
  - Pain

Culture

- Understand why it is important to change
- Empower teams
  - Remove fear of speaking up
- Accountability
  - Delivery is teams responsibility, failure is of the “team”
- Teamwork
  - We work together, we fail together, we win together, no “Hero”
- Learning
  - Share knowledge, pair up and learn together, hack together
- Trust
  - Honesty and ownership
- Reinforce values
  - Award right behaviors, focus on quality

Lean

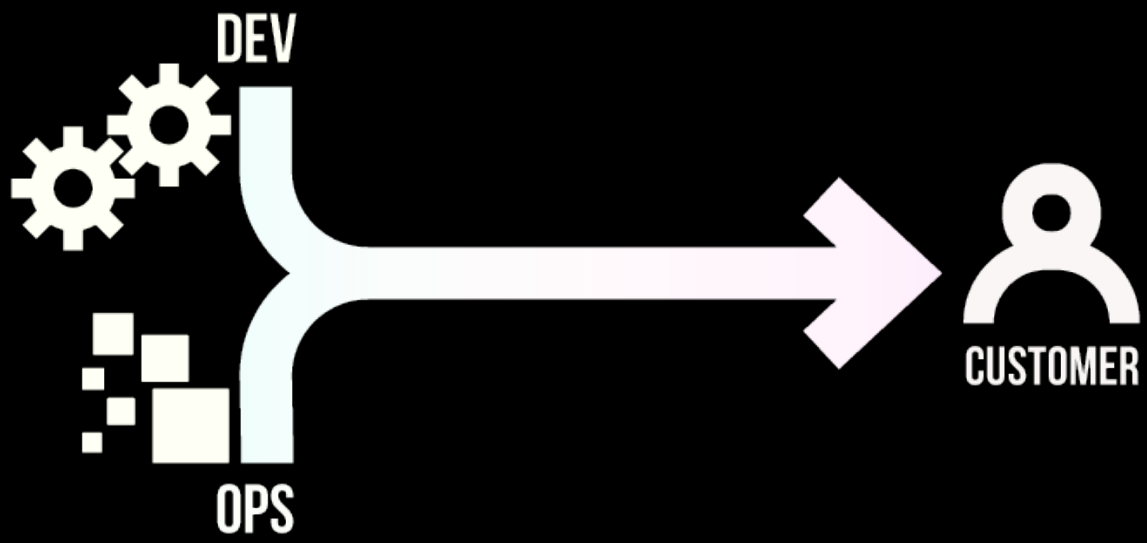
- Focus on customer value
- Eliminate waste
- Reduce cycle time
  - Faster feedback
- Shared learning
  - No silos
- Avoid batching
  - Pull work and do work only when it is required by the next step
- Theory of constraints
  - Remove bottlenecks by using 5 focused steps

Key practices

- Three “Ways”
- Coined by Gene Kim in his *“The Phoenix Project”*
  - First way: Systems thinking – improve “Flow”
  - Second way: Amplify feedback loops
  - Third way: Culture of continual experimentation and learning

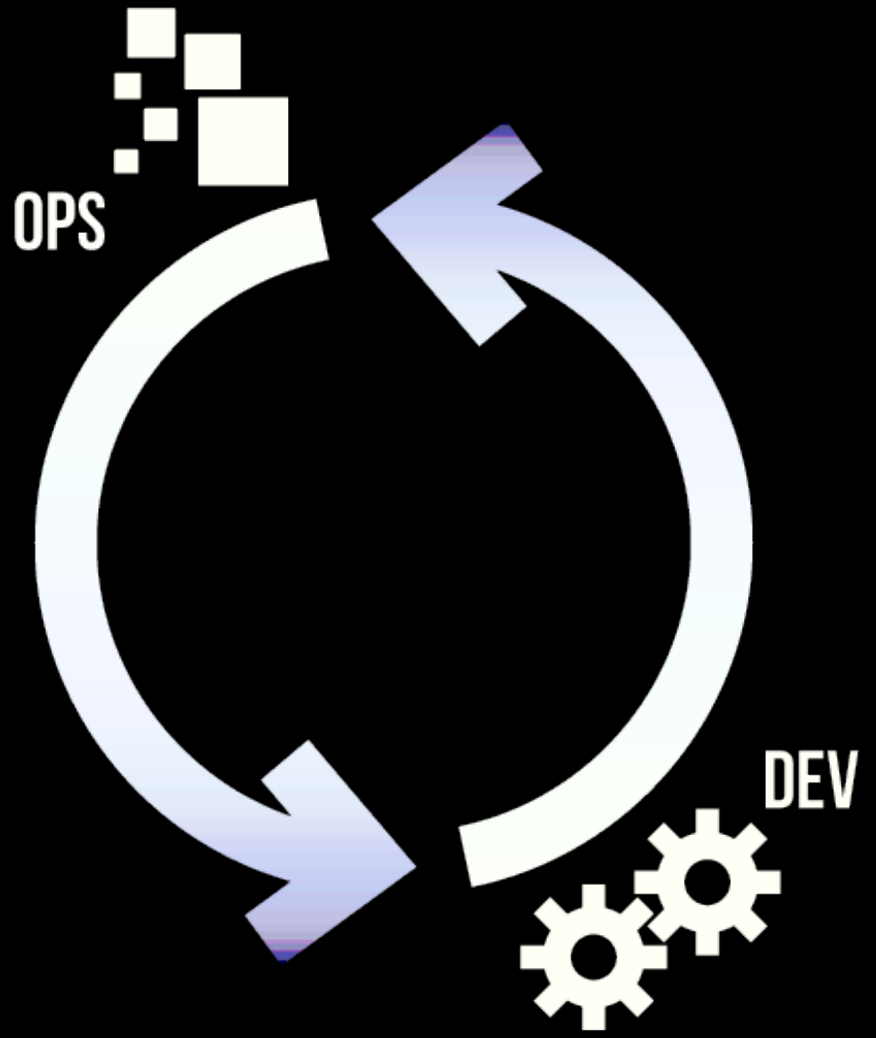
Key practices – Improve flow

- From business to customer as fast as possible
- Lower the amount of WIP
- Identify and remove constraints
- Re-think current processes
- Focus on creating value
- Do not blame people, fix system



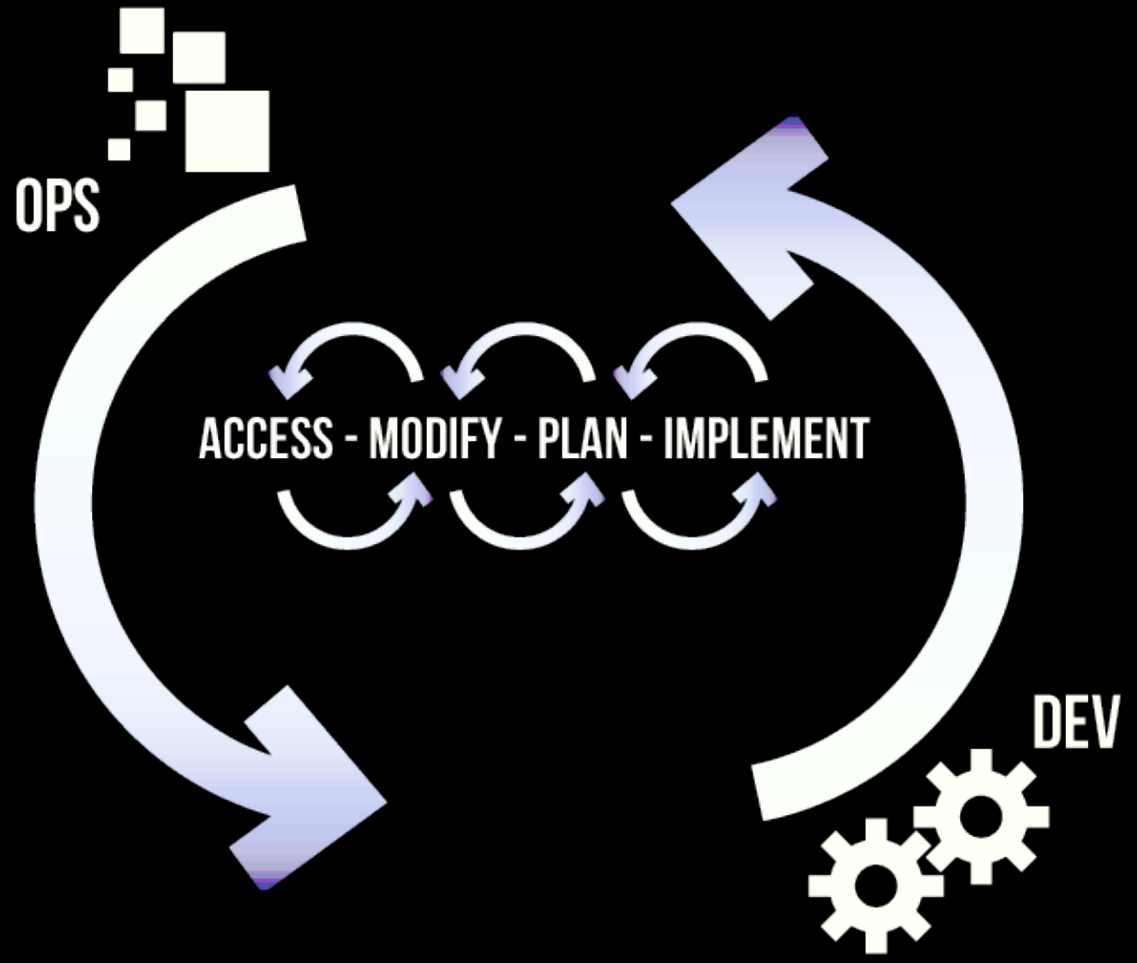
Key practices – Amplify feedback loops

- Establish feedback loops
- Keep feedback loops short
- Concentrate on quality
- Less emotional and monetary investment
- Easy to change
- Improved quality at each stage

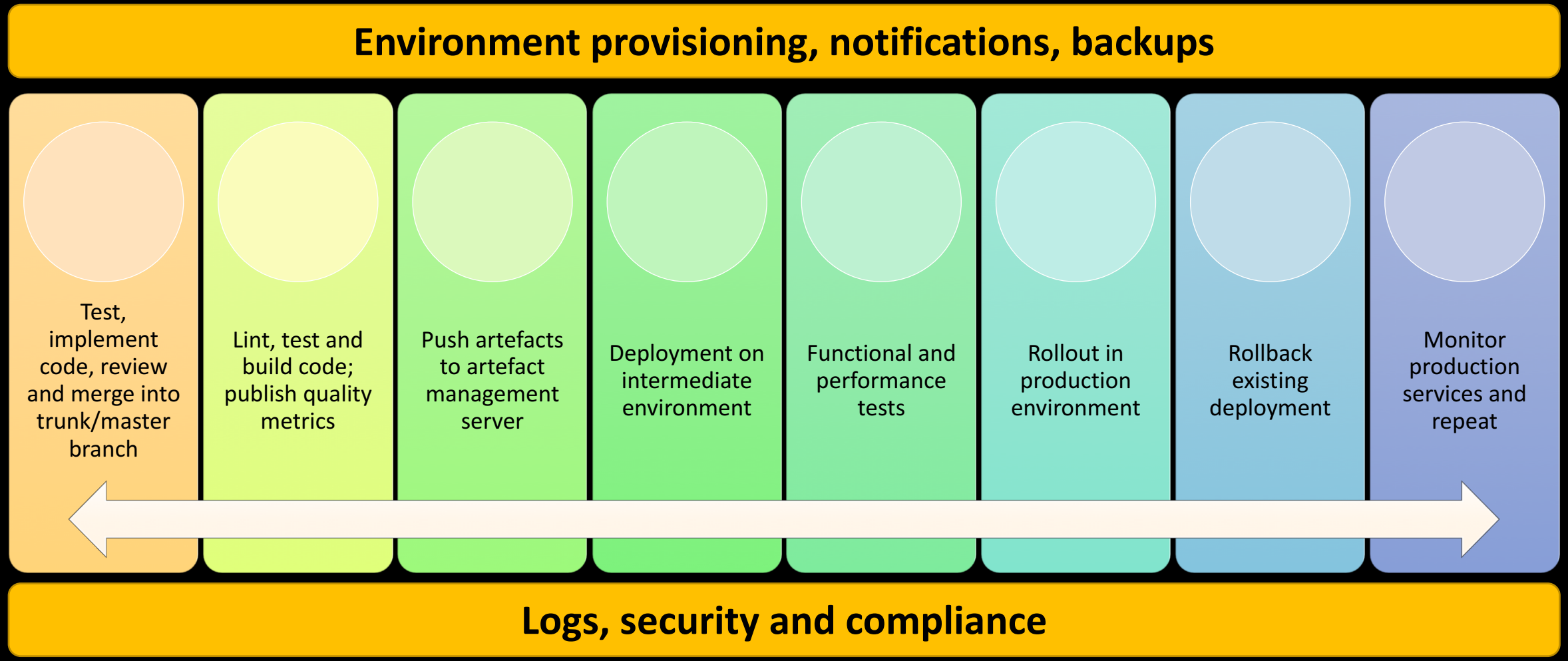


Key practices – Culture of continual experimentation and learning

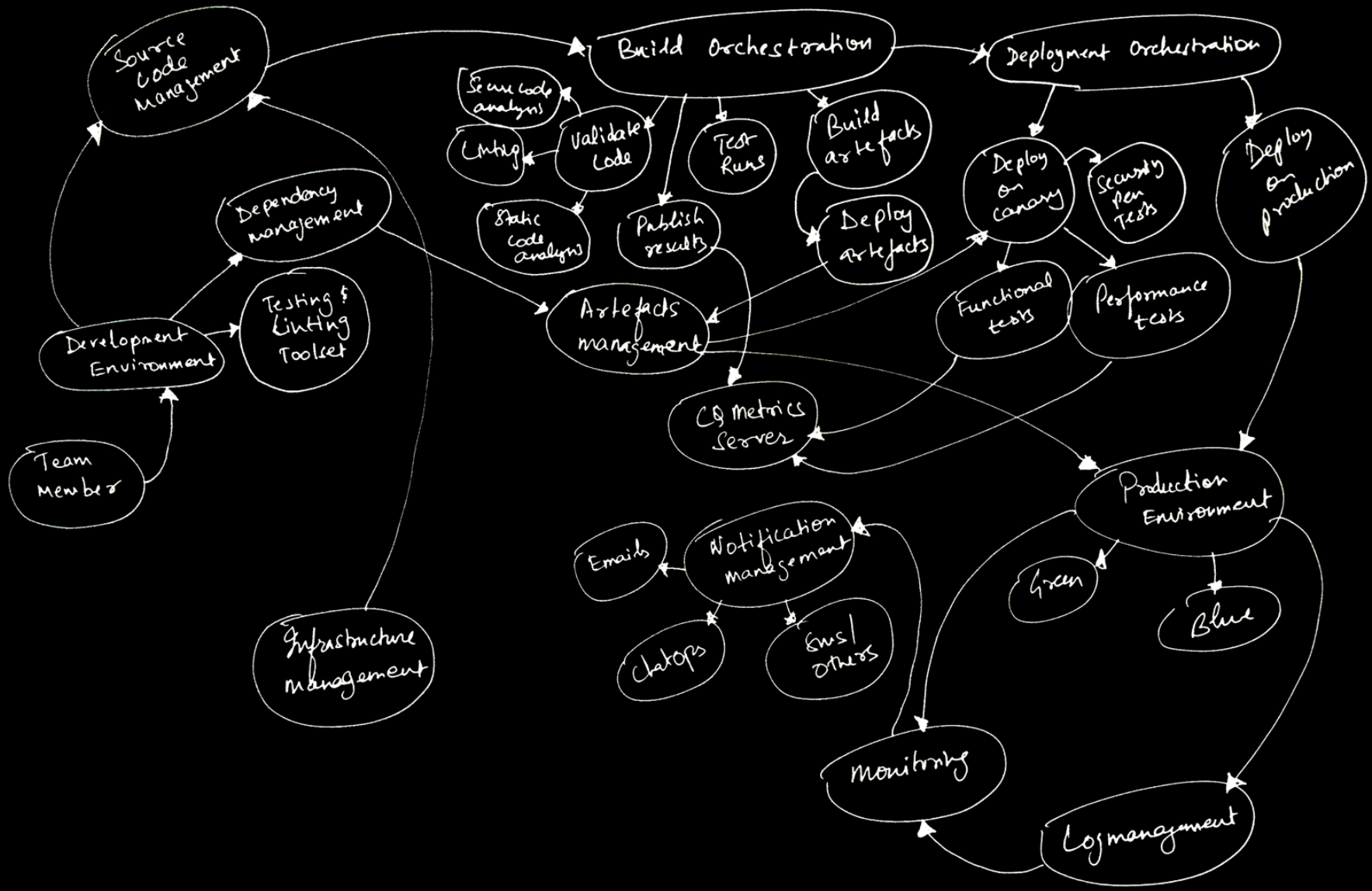
- Experiment and learn
- Foster risk taking
- Minimize cost of risk
- Try new ideas, fail fast
- Learn from failures



Discussion: Typical DevOps pipeline



Typical DevOps pipeline



Hands-on! Development environment

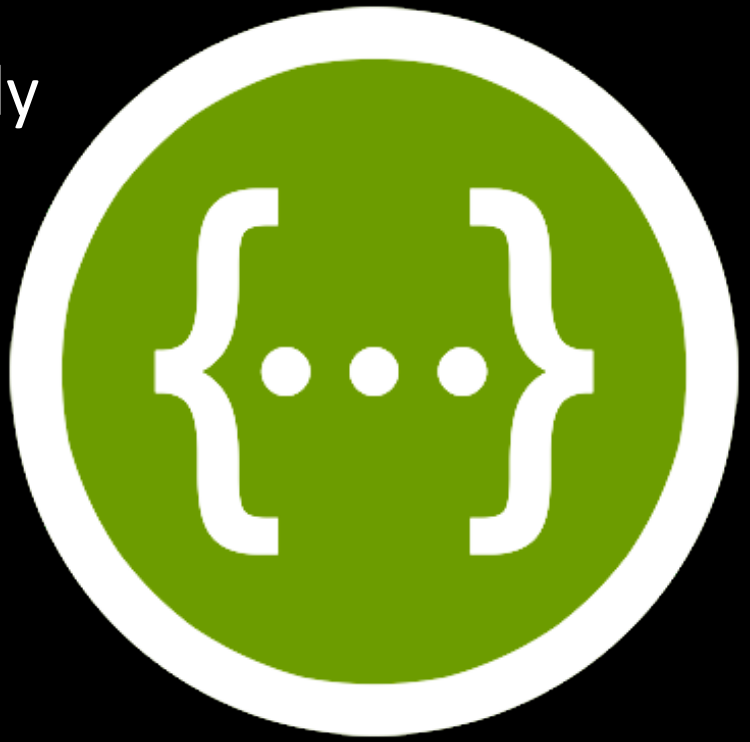
- VirtualBox appliance with all essentials tools pre-installed
- Docker for most exercises/components
- Local docker registry contains most images we require
- VisualStudio code as primary IDE
- MEAN stack with Angular 2

The big shift – GIT

- Lightweight
- Simple to understand
- Distributed
- Super fast
- Agile branching and collaboration
- Easy to maintain and less messy
- Distributed visual collaborative workflows
  - GitHub, BitBucket and GitLab enhance GIT

The big shift – REST (Representational state transfer)

- SOA had a lot of potential
- Lets disparate systems communicate seamlessly
- SOAP had too much over head
  - XML is verbose
  - Frameworks to talk SOAP are clunky
  - Difficult to understand the communication flow
- RESTful services
  - Light-weight (mostly) JSON based communication
  - Easily understood by ay client speaking HTTP
  - Revolutionized API space

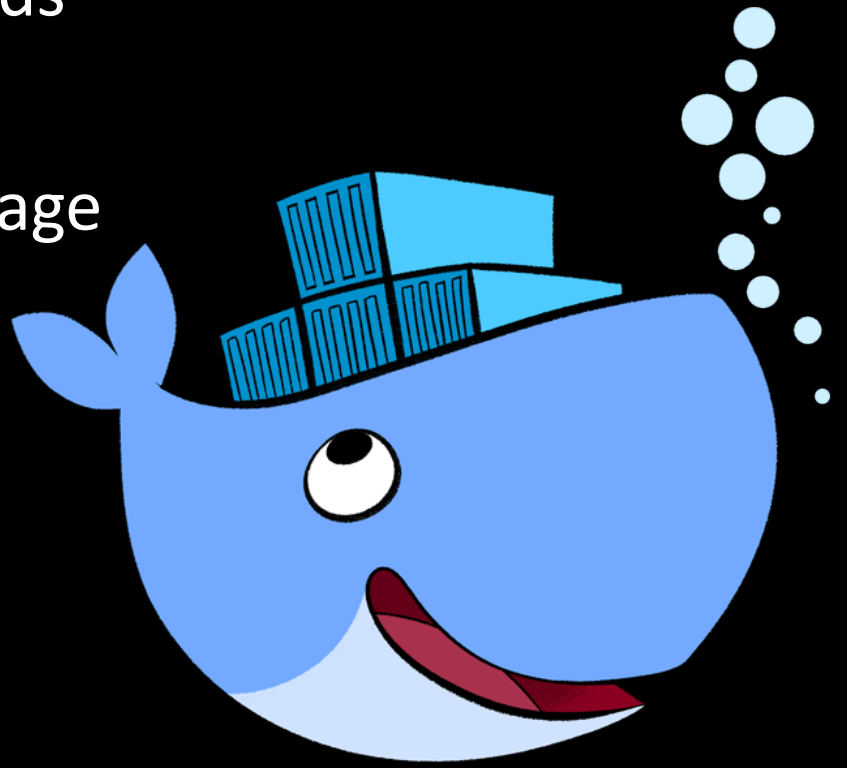


The big shift – Micro services

- Independent, modular pieces of functionality
- Runs on independent process, independent servers
- Can be deployed and scaled independently
- Can be evolved, improved independently
- Smart endpoints and dumb pipes
- Usually talks REST
- Deployment becomes an overhead

The big shift – Containers

- Contains everything that the application needs
- Built on immutable image
- Multiple containers can be spun off same image
- Shares kernel of host OS
- Light-weight as compared to VMs
  - Process isolation using Linux cgroups
  - Does not require the complete OS to run
- Have been around since early 2000's!
- Docker made them accessible to average developer



The big shift – Cloud

- Environment provisioning in seconds
- Pay by the second
- On-demand provisioning
- Dynamically configurable environments
- Full support for IaaS
- Innovative products like FaaS, CaaS, PaaS, *aaS!

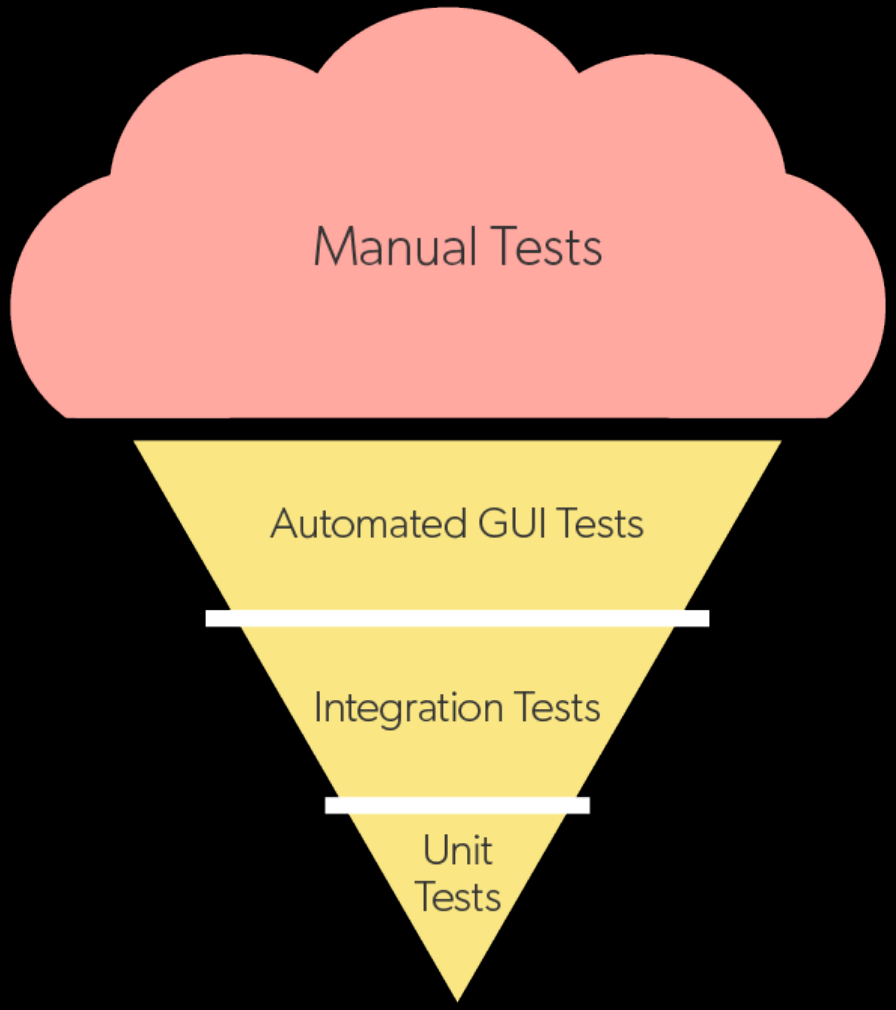
MEAN stack

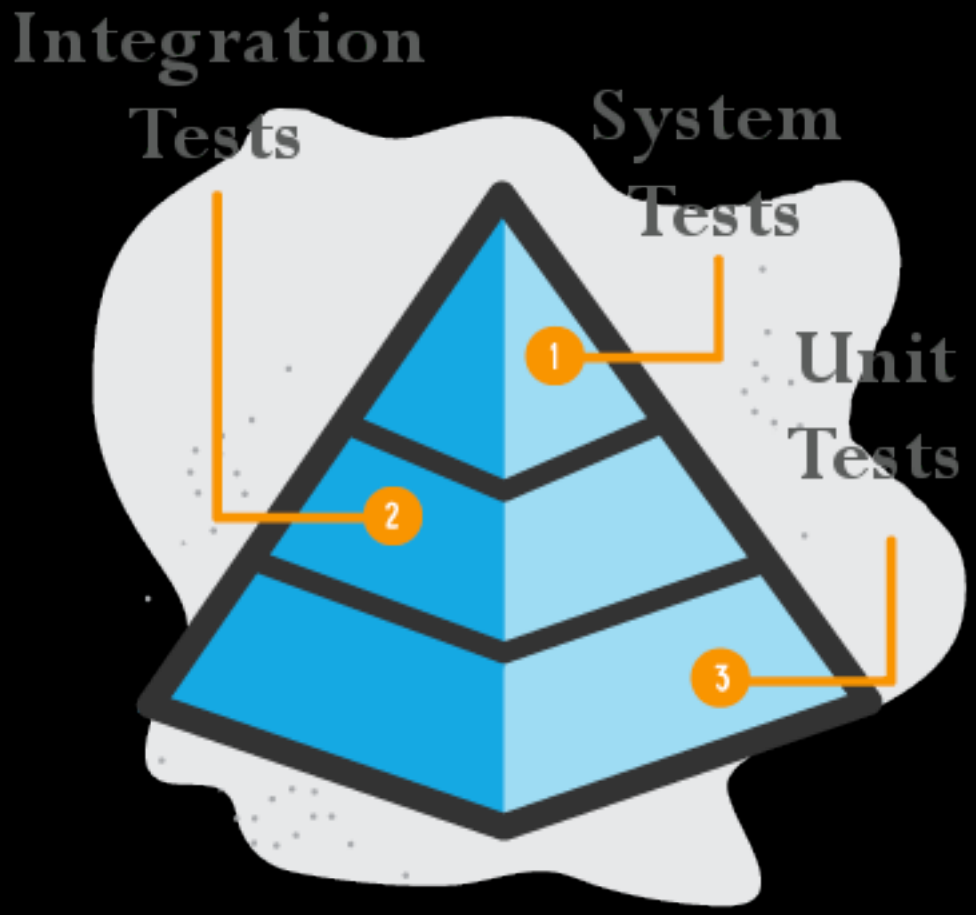
- MongoDB
  - Document database storing document in JSON format
  - Very fast, distributed, scalable (built for web scales)
  - Is a paradigm shift from RDBMS way of thinking
- Express
  - Bring your own server
  - Web server for node
- Angular 2
  - Component based web framework for building complex apps in browser
- NodeJS
  - Platform built on Google’s V8 engine
  - Uses an event-driven, non-blocking I/O model
- Demo of the App that we will build

MEAN Stack Setup

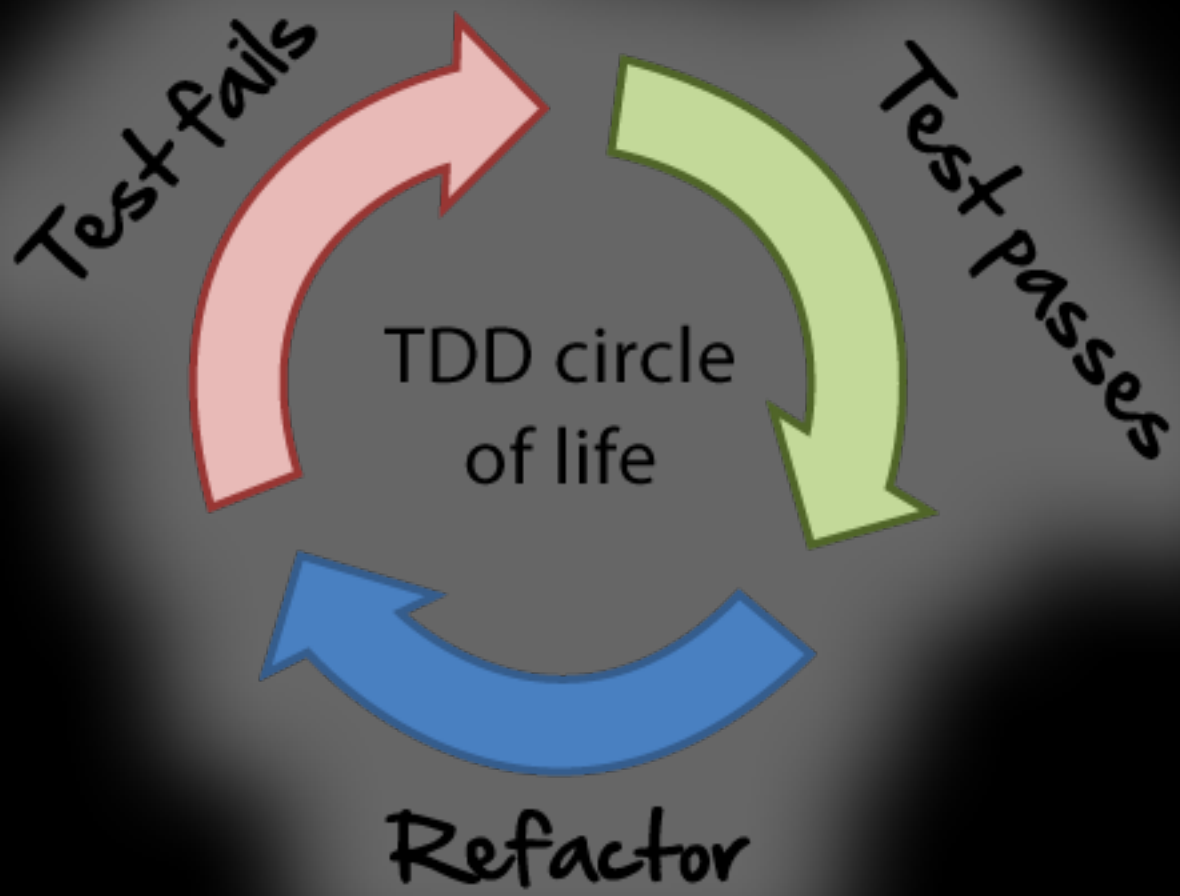
- Angular 2 Setup
  - Different Ways
    - Angular 2 Seed Starter
    - Angular 2 CLI
    - Yeoman Generator
    - Scratch
  - Pre-requisites
- Demo

Ice-cream cone vs pyramid





TDD



Why TDD

- Reduces the level of bugs in code,
- Less application code because we write just enough code to achieve our goals,
- Makes it easier to refactor code,
- Provides sample code of how to use your functions,
- You get a low-level regression test suite, and
- Speeds up code-writing.

- Could give a false sense of quality,
- Can be time consuming,
- Adds complexity to codebase,
- Necessity to have mock objects and stubbed-out code, especially for things outside your control, i.e. third-party code, and
- For a large codebase, tweaking one part of your application, such as a data structure, could result in large changes to tests.

IAAS

- CFEngine
  - Oldest player
  - Built in C, learning curve is very steep
- Puppet
  - Model-based configuration
  - Ruby-based, agent required
- Chef
  - Ruby-based, agent required
- Ansible
  - Python-based, agent-less, configured using YAML
  - Simplest system to understand

Deployment

- Thin line between CI and CD apparatus
- Most CD pipelines can be extended from CI
- Jenkins
  - Orchestrator
  - Building “jobs”
  - Build triggers and notifications
- Artefact management
  - Nexus
  - Artifactory
- Deployment patterns

Log management

- Logs are messy
- Logs are important
- Logs are everywhere
- Cloud environments, containers are ephemeral
- Log aggregators
  - Ingestion
  - Indexing
  - Search, dashboards and alerts
- ELK

Monitoring in production

- Site Reliability Engineers
- Tools to proactively detect issues
  - Nagios
  - Graphite
  - Riemann
  - Prometheus
- Graphing and alerting
  - Prometheus
- Basic container monitoring
  - Portainer
  - cAdvisor

Ops tool demos

- Setup ELK
- Setup Prometheus with Grafana
- Setup log routing from containers to ELK

Stitching everything together

Code features using TDD -> Git checkin -> Jenkins job triggers -> Artefacts created -> Linting and test results published -> Artefacts stored -> Deploy job triggers -> Deployment done on Docker -> Logs piped into ELK -> Grafana dashboards sends updates

Managing containers in enterprises

- Container orchestration
- Docker Swarm
- Kubernetes
- Docker swarm demo

Lets review!

- Learnings
- DevOps – what did we learn - discussion
- Retro

Thanks for attending

Any questions please connect with me on

gaurav.nagar@appworkstechnologies.com

LinkedIn: https://www.linkedin.com/in/gnagar/