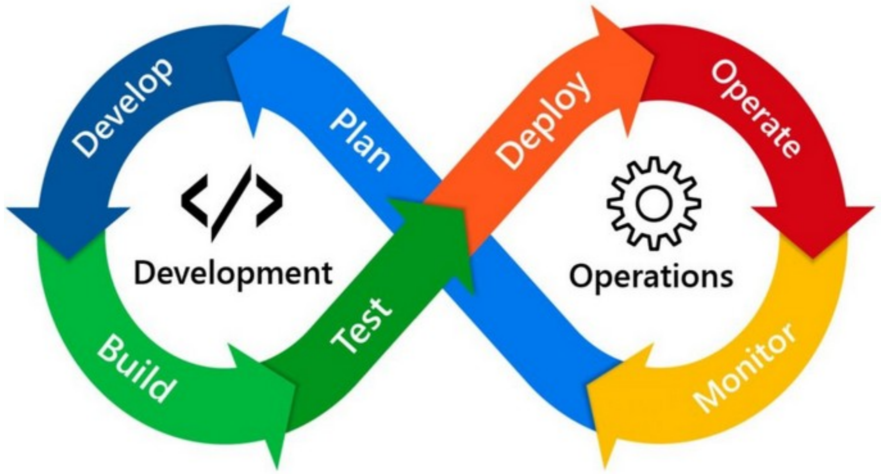
Business Agility

Open Focus: CI/CD

Fortune Buchholtz

The Moebius Strip



graphic credit link

Whut?

- **Continuous Integration/Continuous Delivery-Deployment**
  - **A holistic cultural approach** to software development
    - Emphasising
      - Frequent automated integration of code changes
      - Comprehensive testing
      - Rapid, reliable, small, safe software releases
  - **An organisational mindset change** that
    - Increases customer satisfaction
    - Shortens time-to-value
    - Enables faster, more consistent software delivery
    - Elevates **people**, **communication** & **new Ways of Working**
      - Prioritizes collaboration, quick feedback & incremental improvements
      - Asks that teams work together to integrate code frequently, validate changes automatically

Whut?

It's not just a toolset

OK, So Just Do It?

- It's easy because Gene Kim told us how, right?
  - Create a dedicated transformation team
    - Assemble a cross-functional team with representatives from development, operations, security & relevant departments
      - This team's accountability: driving the transformation & breaking down traditional silos, understand the organisation's Action Logic
        - Set up a transformation Kanban with swim lanes for the work streams, work it
  - Map the value stream
    - Conduct a comprehensive value stream mapping exercise to understand the current software delivery, operational processes
    - Identify bottlenecks, waste, & inefficiencies in the existing workflow from idea conception to production deployment
  - Implement CI/CD
    - Establish automated build, test & deployment pipelines.
    - Ensure that code changes can be rapidly & safely moved from development to production with minimal manual intervention

Yeah, So Easy!

- Keep going…
  - Establish comprehensive monitoring & observability
    - Implement monitoring tools & practices to gain visibility into
      - Tech debt
      - Application performance
      - Infrastructure health
      - User experience
    - Create dashboards & alerts that provide real-time insights into system behaviour & potential issues.
  - Start with a strategic pilot project
    - Choose a representative application or service that's not mission-critical
  - Use this pilot to demonstrate potential benefits
    - Generate early wins
    - Learn valuable lessons
  - Celebrate the wins, review, retrospect & improve
    - Rinse & repeat until the angels sing

YES, BUT NO!
SORRY.

Sight Our North Star, Name Our Outcome

Sight Our Star Together

- Articulate how CI/CD takes us all towards our North Star
  - Co-create with leaderful collaboration on a transformation team
    - Ask for volunteers, watch the change agents pop up
    - Widen the group for co-creation & diverse perspectives
      - Invite those representing customers, vendors, business, security, architecture, IT & customer services, finance...
    - “Nothing about you without you”
    - Explore what capabilities we need & what must we evolve to develop them
    - Welcome constructive challenge
  - Consider the Liberating Structure Strategy Knotworking for Sense-making
    - Alternate strings could include Nine Whys, Draw Together, Ecocycle Planning, Purpose to Practice, Min Specs, 15% Solution...
    - Bring data & facts to bear, talk about money, talk about value
  - Choose our metrics carefully in light of our system & culture
    - Larman’s Laws are real & have big teeth
  - Sense-Respond-Make-Sense our way forward towards a new pattern
    - Our knowledge remains partial
    - “Learning is better than mere knowing.” - Amy Edmondson
      - “Live the questions.” - Rilke

Prepare People for the Voyage

- Communicate & socialize our North Star
  - Hear people's voices with openness & curiosity: they are the experts
    - Conway's Law: our communication structure shapes the systems we design, including our CI/CD
  - Their feedback is our gift & adjusts our course
  - Humbly invite their participation, acknowledge the J-curve
- Address the state of psychological safety
  - Survey, accept & socialise the results
  - Formally commit to improvement at every level: write it into HR policy, make it bonus-able for management
  - Improve it by modelling vulnerability, humility, transparency, trust, collaboration & accepting feedback
  - We anchor the holding space in which safety can blossom widely to change the environment
- Let people evaluate their current skills & identify knowledge gaps around CI/CD practices
  - Implement a continuous learning strategy with the People team & leaders, if we don't have one
    - Where can we adopt formal HR processes to require regular learning & improvement?
      - Lunch & learns or Friday book clubs in the interim
    - Develop formal space for practice communities to support ongoing learning & skill development

Address Our Obstacles

- Openly ask: how can we address our immediate obstacles?
  - Lack of psychological safety
    - Belinda's external training, vertical facilitation for leadership & managers to elevate the Action Logic
    - Poor communication
      - How does our current structure impede our communication?
      - How can we improve our Ways of Working to emphasis more communication?
    - Cognitive overload
      - What ideas from Team Topologies could fit our context?
  - Frequent hotfixes
    - How can we “be quick but not hurry”?
    - How can we make our changes smaller, safer?
    - What could TDD or ensemble work look like for us?
    - How can we automate more build & test processes?
  - Manual testing bottlenecks
    - What would more efficient UAT look like?
    - What would configuring deployment pipelines to ensure consistent, reliable releases look like?
  - Staging environment issues
    - How can we use Staging better?
  - How can we adopt Infrastructure as Code?
    - What skills & tools would we need to manage infrastructure changes consistently?

Expand Our Structure

- Foster Cross-Team Collaboration
  - Open the gift of purpose, autonomy, ownership
    - Emphasize that we are all accountable together for the quality & successful delivery
      - This includes the business, portfolio, managers & leadership
      - Model accountability to & for each other
  - Smash the silos, break the frames
    - Reward collaboration between Dev, QA, Ops & relevant teams
      - Prioritise this discovery & learning phase
    - Introduce the concept of value streams now
      - “From the customer back to the idea” & business capability
      - Assure everyone they can co-create the stream
      - Hold the space where people tell us what needs to be done
  - Involve Non-Engineering Teams
    - Include design, product, marketing, sales, compliance, IT service, vendor & customer services, etc.
      - Profit from their institutional & customer knowledge
      - Let them be heard

Focus on Value Creation

- Openly ask: what does excellence look like for us?
  - Everything is on the table for review & improvement
    - Do our current Ways of Working & their structures result in better communication, better value delivery?
- Inventory our stuff!
  - How much stuff is in the system?
    - Sort by risk
  - Scrutinise our stuffs' value & relationship to value
    - This means employee happiness, customer satisfaction, business capability & money
  - Ruthlessly remove non-valuable work to improve focus
    - Kindly skip that meeting, that email, that phone call
- Count how much work is left!
  - Openly ask: how much work can we really do well at this moment?
    - What would Work-in-Progress limits look like for us?

Towards Technical Excellence

- For Devs/Test/Release:
  - How can we improve version control practices?
  - How can we further improve automated build & test processes?
    - Consider improved unit, integration, functional, API, performance & security tests
    - Continuously improve deployment pipelines to ensure consistent, reliable software release
  - How can we adopt a micro services architecture?
    - Consider using micro services to support agile, scalable delivery across different environments
  - How can we implement feature flags?
    - Consider feature flags to decouple deployment from release, allowing for more controlled and gradual rollouts.
  - How would other roles in our process towards the customer answer these questions?

Measure Then Optimise

- Openly ask: How do we measure & optimise?
  - Beware Goodhart's Law!
  - Implement value stream mapping:
    - To visualise our holistic workflow
    - To identify bottlenecks
    - Continuously improve our entire CD flow & pipeline
  - Define more metrics
    - Lead time, cycle time, work item age with Monte Carlo where appropriate
    - Deployment frequency, lead time for changes & mean time to recovery
    - Where can we measure customer & business value?
  - Regular review & improvement:
    - Continuously assess our CI/CD experiments
      - On feedback, performance, Ops data
  - Step back to ask ourselves
    - Have we changed our culture, built people, clocked faster or more reliable delivery, improved productivity, & increased employee happiness?

Keep Experimenting

- Keep asking: what does excellence look like for us?
  - Openly ask: are we even doing valuable work?
  - Investigate what stops us from doing valuable work
  - Experiment with how we can remove these obstacles to do valuable work
    - Maybe writing clean maintainable code / processes / requirements / compliance guides / tickets / work orders, etc.?
    - Maybe implementing rigorous testing from the start?
    - Maybe automating testing?
    - Maybe never letting a problem defect or issue go forward?
    - Maybe fearlessly raising issues, all debt & customer problems?
    - Maybe learning to say, “No, not now”?

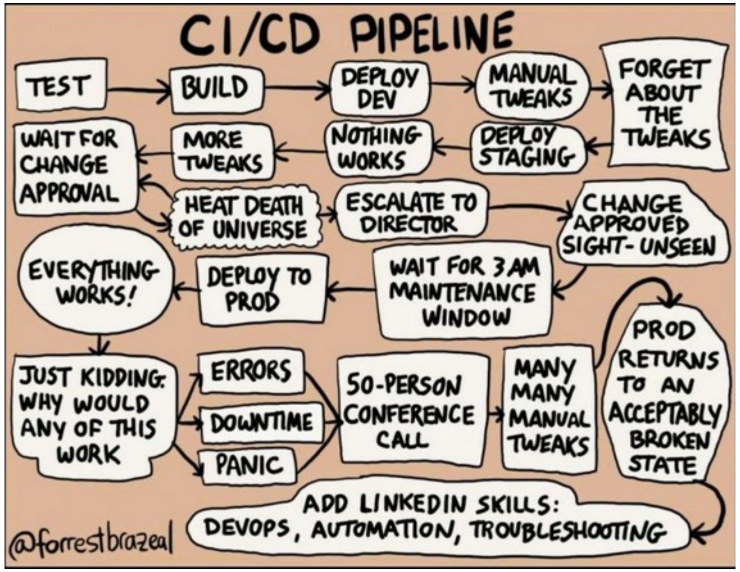
NOW DO THE GENE KIM

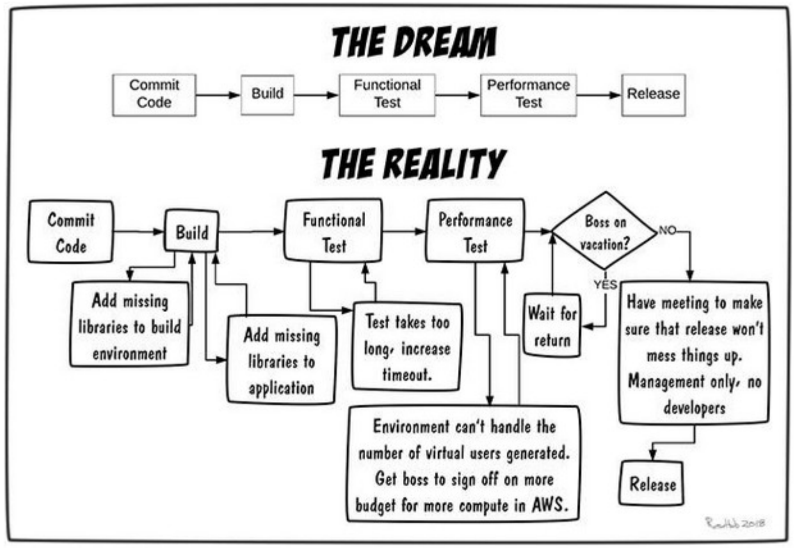
because we have our outcome, our meaning & our context

Those Sea Monsters are Anti-Patterns

- Common antipatterns
  - Scatter-gather
  - Poor communication
  - Waiting to commit code until the entire feature is complete
  - Few builds per sprint or per week
  - Non-relevant branch name or unclear commit messages
  - Too much manual code testing after packaging or deployment
  - Testing directly in production
  - Restricting access & documentation to certain teams
  - Not collaborating or helping each other
  - Drifting back to old habits due to distractions or pressure

Those Sea Monsters are Anti-Patterns





THE DREAM
Commit Code
Build
Functional Test
Performance Test
Release
THE REALITY
Commit Code
Build
Functional Test
Performance Test
Boss on vacation?
NO
YES
Add missing libraries to build environment
Add missing libraries to application
Test takes too long, increase timeout.
Environment can't handle the number of virtual users generated. Get boss to sign off on more budget for more compute in AWS.
Wait for return
Have meeting to make sure that release won't mess things up. Management only, no developers
Release