We The Few

Augmenting the Continuous Delivery Team With Automation

@keith_strini
@dmfrey

THE LINUX FOUNDATION

Who Are We?

Dan Frey…

Advisory Solutions Architect on the App Transformation team at Pivotal. I help companies move their legacy workloads to the cloud. I work hands on with your employees helping them replatform and modernize your apps and instruct them on XP and lean principles.

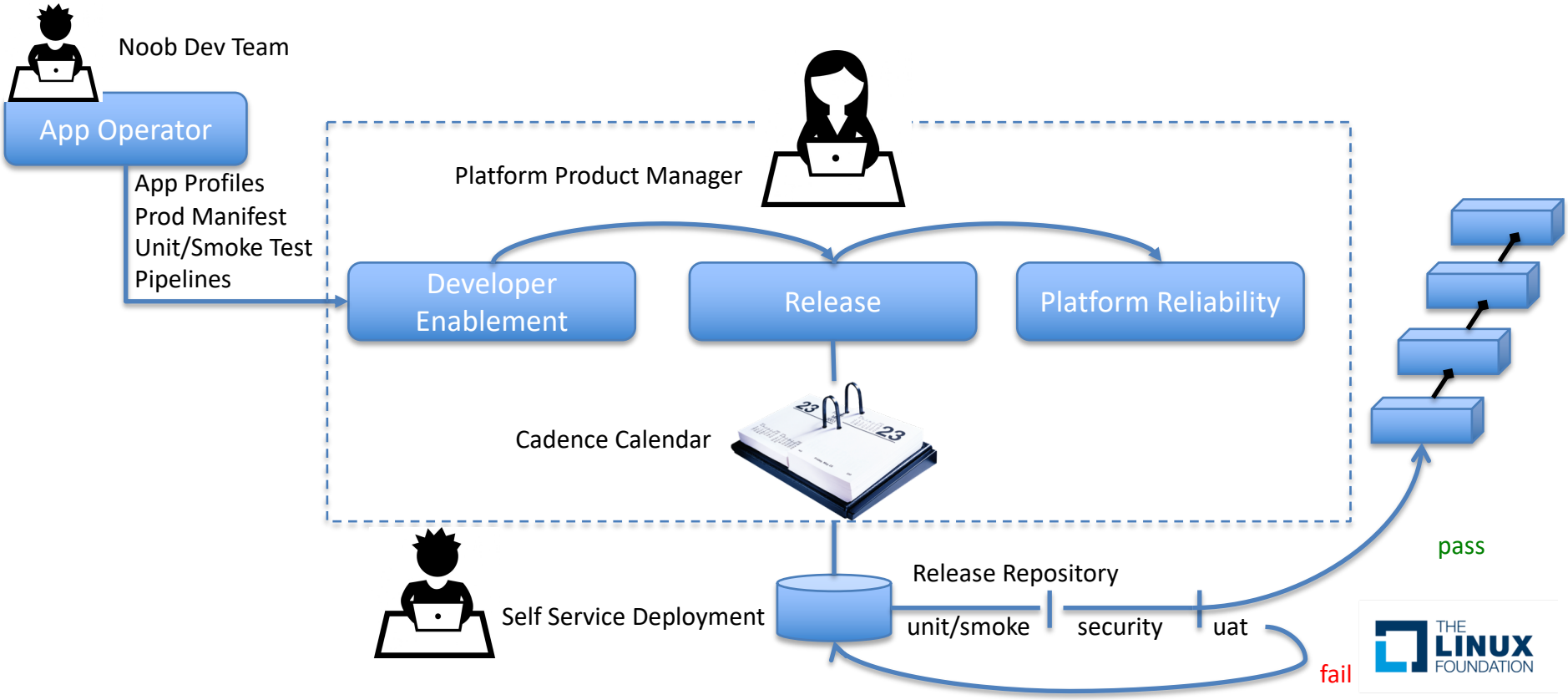
Keith Strini …

Field Facing Solutions Architect that serves as a technology analyst for the US Department of Defense and Intelligence communities. I architect, develop and field information systems across the Joint Services both CONUS and OCONUS (Korea, Japan, Europe, and the Middle East) and NATO

THE LINUX FOUNDATION

End Vision vs Starting Vision

Release Engineering Stratification



Developer Enablement

Coordination Point For All New Development Efforts.

PRACTICES

- Creates/Coordinates Platform on boarding meeting
- Provides Latest Information about Platform Environments

Release Engineering

Coordination Point and Execution Lead For All New Releases.

PRACTICES

- Attends Final Pre-Release Demo of Apps
- Verifies Release Artifacts
- Coordinates Initial Release Date
- Collaborates on Downstream Environment Triage

Platform Reliability

Coordination Point For All Platform Environment Changes.

PRACTICES

- Creates/Coordinates Cadence Meeting
- Continuously Develops Resiliency Probes based on Post Mortems
- Maintains Environment Parity
- Enforces Strict Runtime Version Control
- Communicates Environment Adversity
- Creates/Coordinate Resiliency Exercise
- Instruments Distributed Tracing in Ops

THE LINUX FOUNDATION

Failure is Inevitable, Hope is Not a Strategy

* Complex System Failure

- Changing mixtures of failures latent within them
- Post-accident attribution accident to a ‘root cause’ is fundamentally wrong.
- Change introduces new failure
- Views of ‘cause’ limit the effectiveness of defenses against future events
- People continuously create safety
- Failure free operations require experience with failure

** Simple System Failure

- Simple systems fail
- Simple systems fail less often than complex systems
- Simple systems fail predictably
- Simple systems fail transparently
- Simple systems fail affordably
- Simple systems fail educationally
- Simple systems fail by being inadequate
- Simple systems fail by becoming complex

Simple Complexity

- Create complex behavior from simple building blocks
- Maintain simple independently testable components
- Offload Management of complex orchestration
- Inject fail safes ‘between’ simple components
- Unknown failures are then a result of ‘rare’ emergent behavior

When you think of testing…

Unit Testing… That's so Justin and Britney…

Smoke … pfft … we couldn't come up with a software based term?

Bringing Sexy Back – Other tests don't know how to act

Developers	Dev Enablement	Release	Operations	Operations
				
Unit/Smoke	**Contract**	**Feature Flagging**	**Canary**	**Distributed Tracing**
Been There, Done That, Got the swag!	**Focuses Developers on Contract Based Testing For Integration**	**Decides on Feature Maturity From a Stability Perspective**	**Care And Feeding Of New Releases To Ensure Early Intervention**	**Deep Understanding of How New Efforts Deploy Into Operations**
Traditional Unit Testing Frameworks	Spring Cloud Contract Framework	Smart Platform Environment Variable Injection Coupled with Dynamic Routing	Dynamic Instrumentation Out Of The Box Coupled with OSS Production Non-Functional Libraries	OSS Production Libraries Coupled with Production Platform Monitoring/Metrics and Threshold Alerting integrated into Logging, Monitoring and Notification Services

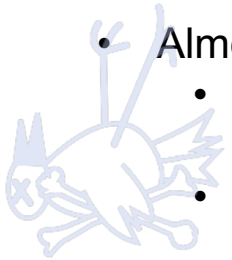
Decoupled Integration

- If Speed Is What you Want, End-To-End Testing is not how you get there.
  - Getting feedback…this week?
- Are you mocking me?
  - Single Source of Truth…
  - Verifying the Goods
- Isolation testing of Single Services (Provider or Consumer)
- I do not think that means what you think it means (Semantic Testing)
- Complexity From Simplicity, Not Complexity From Complexity
- Test Data…let's not ignore the elephant in the room
- Stability, Stability, Stability… we're talking Operations not Science Experiments
- Ah Sunsets…
  - Paying off Technical debt by subtraction *and* addition
- You get me… you really get me
  - Consumer defined APIs

Maintaining Operational Velocity

- I dunno. You tell me what you want.
  - Non Techs getting in on the Action
- Ok so most of it works but I gotta send it back?
  - Beauty of context encapsulation
  - Waiting for a feature like you
- I see how we do 1 app but how do I manage 1000s?
- So what if we don't know exactly what our users want?
- Ah Sunsets…
  - Paying off Technical debt by subtraction *and* addition

Predictive Fire Fighting in Operations



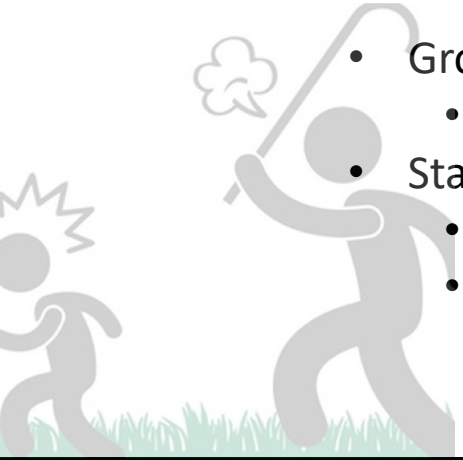
- Almost trust you
  - Canaries – Profiling the CPU, memory, disk usage, cache synch
    - Rollback/Roll Forward Strategies
  - Blue/Green Deployments
    - The case of stateless
    - The case of stateful (transactions, migrating data)
    - Infrastructure Isolation
  - A/B testing
- It not you, its me
  - Distributed Tracing
    - Yes we are talking scale here
    - But that's a lot of instrumentation
    - Correlation is tough
      - Good definitions of SLO/SLIs
      - Threshold tuning

Operations as The Caretaker of Code?

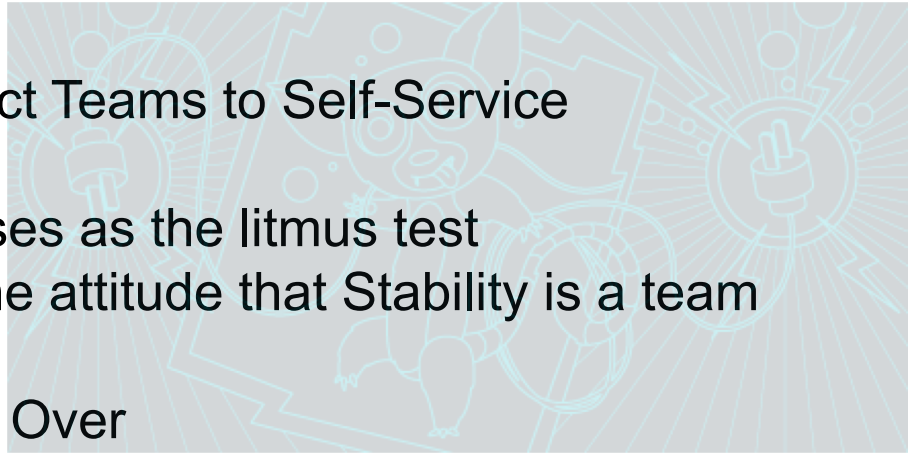
- Your baby is ugly, Our baby is cute
  - Platform as Product helps align our interests
  - Automation helps us be responsive as a team to our end users
  - Lot's of up front pain is better than chronic pain indefinitely
  - Success as defined by rhythm

Get Off My Lawn!

- Change Inherently Creates Failure
  - Alignment of Values
- One Team One Fight
  - Joint SLOs
    - Platform SLIs
    - Application SLIs
    - Instrumentation
- Growing up is hard to do
  - Graduating Product Teams to Self-Service Deployments
- Starting the Cycle Over
  - Capturing Lessons Learned from every class
  - Knowledge Transparency aides the greater team



Growing Up Is Hard To Do



- Graduating Product Teams to Self-Service Deployments
- Resiliency Exercises as the litmus test
- Communicating the attitude that Stability is a team sport
- Starting the Cycle Over
  - Capturing Lessons Learned from every class
  - Knowledge Transparency aides the greater team

In Conclusion

“Sometimes you have to practice going slow, if you want to eventually go fast forever”

- End Vision – You can manage operations at scale with very small light weight teams
  - Release Engineering Maturity is a fundamental piece of that vision
  - Understanding your particular environment’s adversity takes time
  - Team chemistry takes time
  - Automation takes time
  - Take your Time. Your team as a whole will operate better because of it.

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

THE LINUX FOUNDATION