ticketmaster®

DOUBLING DOWN ON DEVOPS

JUSTIN DEAN, SVP TECHNICAL OPERATIONS

DEVOPS ENTERPRISE SUMMIT
LONDON 2016

ABOUT

- Justin Dean - SVP, Platform & Technical Operations
  - Joined Ticketmaster in Feb 2015
  - Previously with Shopzilla/Connexity
- 155 TechOps Team Members Globally
  - Infrastructure
  - Platform
  - Network
  - Database
  - Systems Engineering
  - Service Delivery
  - Incident Management

ticketmaster®

HISTORY

- **Publicly Traded Company (LYV)**
- **$7.6B Revenue, $25B in GTV**

- **1976 - Founded at Arizona State University**
- **1996 - Ticketmaster.com launched**
- **2010 - Live Nation and Ticketmaster join forces to power live experiences**
- **2011 - Transformation journey begins...**

ticketmaster®

WHAT WE DO

We Connect Fans to Great Experiences!

Each year our platform powers:

- Over 26,000 Live Nation events to millions of fans worldwide
  Sports, Concerts, Festivals, Theater, Arts, etc.
  Every 20 somewhere in the world there is a Live Nation event!
- Over 53 Million fans in over 40 countries
- 530 Million Ticket transactions
- Own and Operate 167 Venues
- More than 1B unique visits to our web front
- 60% of our traffic is mobile (app + mobile web)

Ticketmaster is a Top 5 ecommerce site

BIG SCALE, BIG CHALLENGES

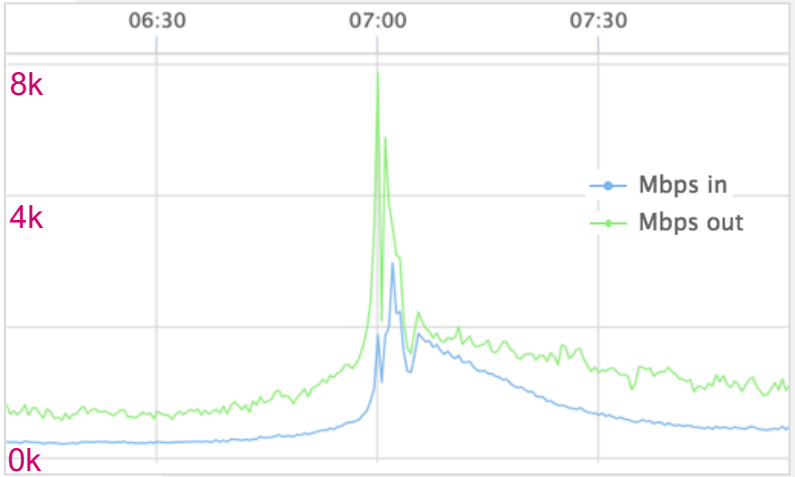
- **Onsales** = *Black Friday* traffic <u>every Fri & Sat</u>

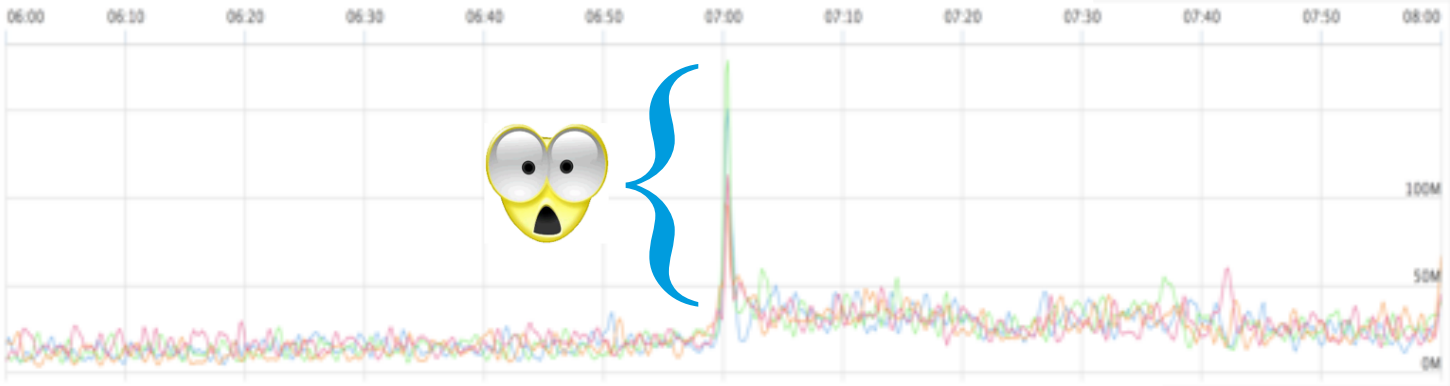
  Huge spikes / demand for tickets

  Limited inventory

  Adele! 10 million fans, only 400K tickets
- 0 to 40MM transactions <u>in minutes!</u>

That's a spike of **>8 GBps !!!!!**





OUR UNIQUE ENVIRONMENT

- Global company distributed in 22 Countries
- Huge tech stack accrued over 40yrs. Plenty of juicy technical challenges:
  - 1970s: Custom VMS OS on Emulated VAX (The Host)
  - 2000s: Big-iron filers, NFS, Xen Cloud, modperl, custom built infra (cache)
  - 2010s: Public Cloud, Docker, Terraform, API's, Microservices
- Custom Private Cloud with over 22,000 VMs across 7 global data centers
- Over 15,000+ network endpoints across the world (Venues, Arenas, etc.)
- Over 60% VM growth in last year
- Keeping pace with this growth is very challenging

TICKETMASTER PRODUCTS

21 Ticketing Systems and over **150** unique products!

- More Products than all of our competitors combined
- Fewer People PER PRODUCT than our competitors
- ~1:1 ratio of Products to TechOps team members

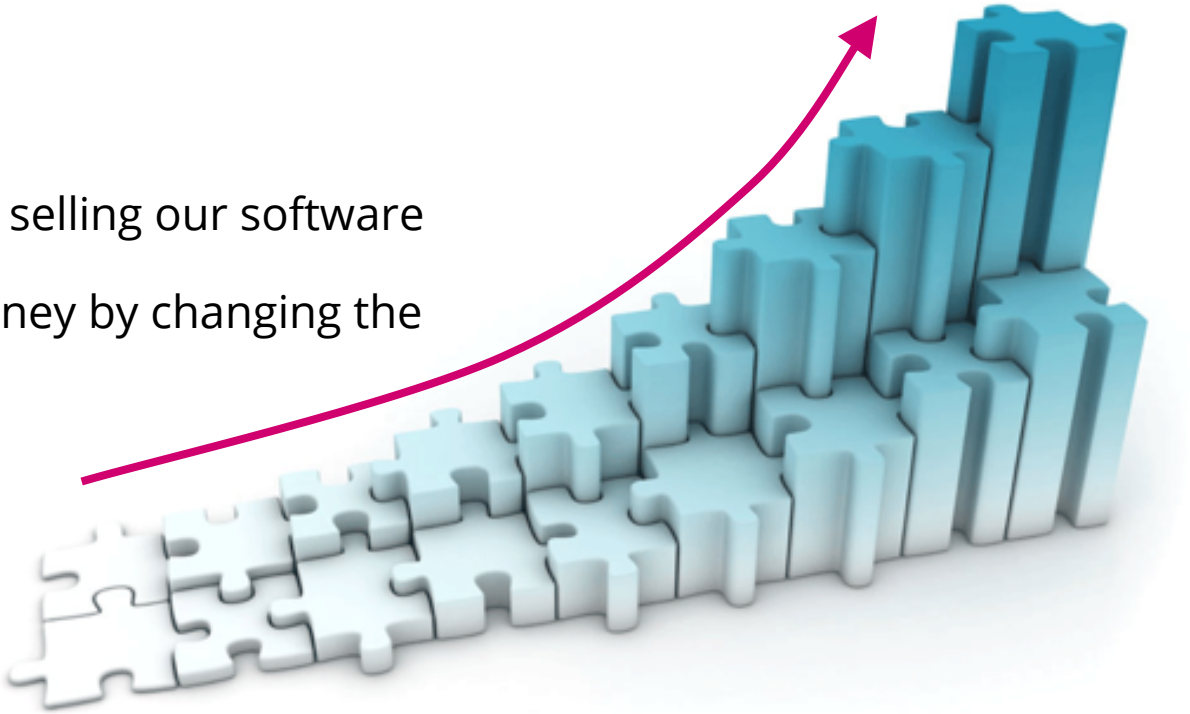
=

Complexity at scale!


ticketmaster®
COMPETITIVE
PRESSURE

2016 BUSINESS PRIORITIES

- $7.6B rev in 2015, $25B in GTV,
- Market Pressures (12% growth, 34% growth in Secondary Market)
- 40yr old SaaS:
  - We only make money by selling our software
  - We only make MORE money by changing the software

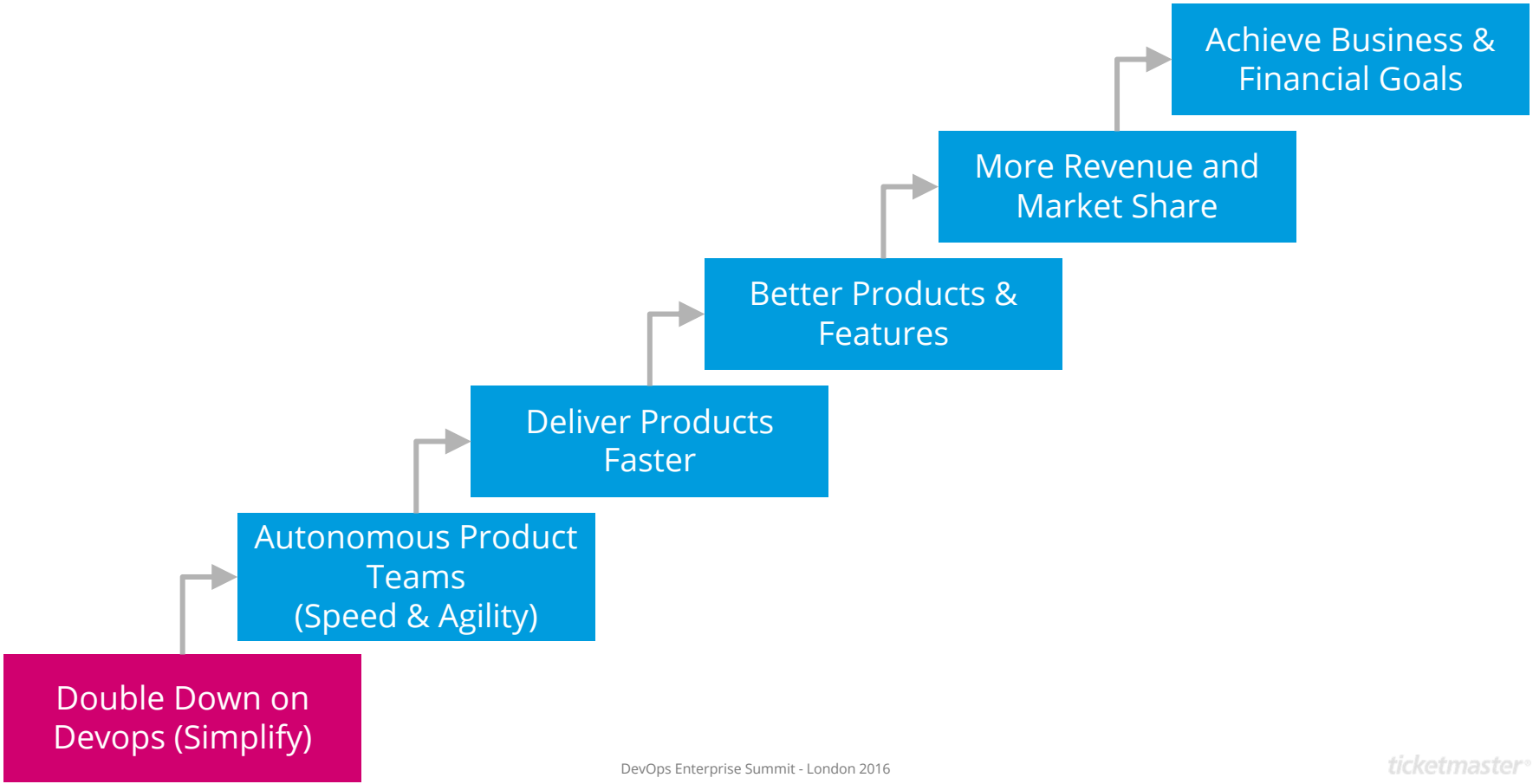


COMPETITIVE LANDSCAPE

- Market leader with huge surface area
- Gravitational pull of established business make rapid changes very hard
- We have everything to lose, competition may have everything to gain
- Competition from every angle, every size and shape
- Speed and Agility are absolutely essential


MUST
GET
FASTER!
ticketmaster®

HOW DO WE GET FASTER?




ticketmaster®
DOUBLING
DOWN

OUR JOURNEY

- **Where we started:**
  - 40 yr SaaS company with deep rooted OnSale culture and high silos
  - Dev dev'd it, Ops operated it.
  - Waterfall and not able to keep pace
- **We Devops'd! (Iteration 1):**
  - Lean Transformation across company
  - Cross-functional teams
  - Automating, Build-Pipelines, Metrics
  - Started blurring lines between Dev and Ops

CHALLENGES

- Culture around “OnSales” - Heavy Ops ownership, prevented true shift of responsibility to the product teams themselves
- Focused on output, not business outcome and value delivery
- Teams were highly siloed by functions and skillset
- Expanded engineering teams by 230% and didn’t scale ops capabilities
- Ecosystem complexity staggeringly high
- “Blocked” and “Friction” were still the norm

DOUBLING DOWN

ITERATION 1 (THEN)	ITERATION 2 (NOW)
Cross-Functional Teams	Autonomous Product Teams
Ad Hoc Tribal Knowledge Sharing	DevOps Training Program
Lean Transformation	Value Driven Delivery
Talking the Devops Talk	Walking the Devops Walk (Measured Maturity Models)
Are We Winning?	Scoreboards

[THEN] CROSS-FUNCTIONAL TEAMS

- Created 55+ cross-functional delivery teams to eliminate dependencies but only included Product/Dev/QA/UX; no ops team members

but then…

- Created world's largest teams: fully cross-functional team, all disciplines, 30+ people

LESSON:

Full-stack teams ≠ fully staffed teams

(Everyone brought their silos with them)

ticketmaster®

[NOW] AUTONOMOUS PRODUCT TEAMS*

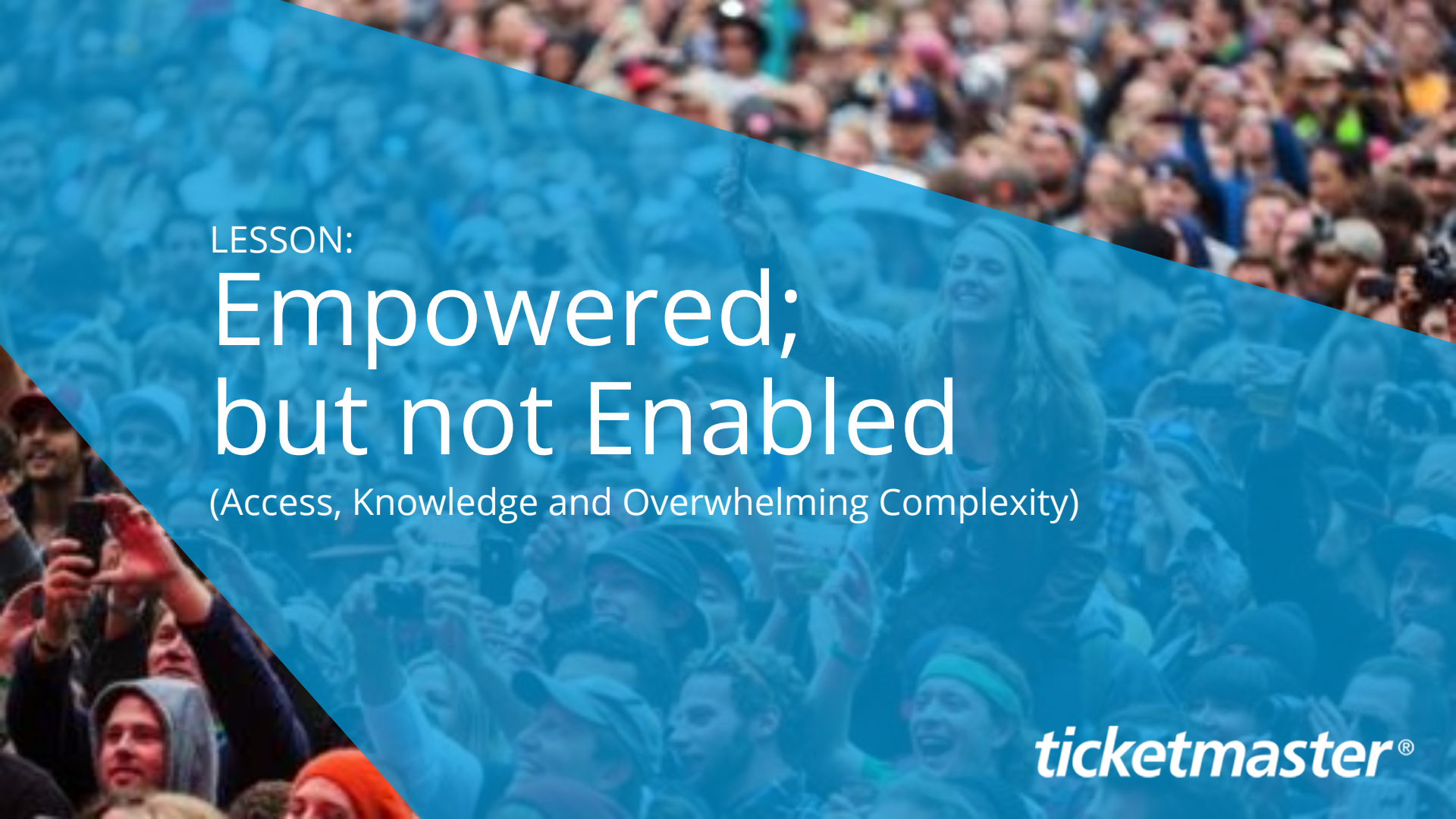
- Transitioned Primary onsale support to product delivery teams
- Moved application support teams out of TechOps and into the product teams directly (Core Ticketing, Payments, etc)
- Embedded Systems Engineers into product delivery teams - “*ility Partners”
- Self-Service Tools - Surge towards getting teams out of the ops business
- Self-Sufficient businesses (build it, run it, own it, optimize it, monetize it)

** Requires org changes*

[THEN] AD HOC KNOWLEDGE SHARING

- Highly Siloed Knowledge
  - Software Development, Systems, Network
  - Product, App Support, Database, etc…
- Product Knowledge != Operational Knowledge
- Heavy dependency on external teams

*** EVERY TICKET IS AN ERROR CONDITION!**



LESSON:

Empowered; but not Enabled

(Access, Knowledge and Overwhelming Complexity)

ticketmaster®

[NOW] DEVOPS CROSS-TRAINING

- **GOAL**:
  Enable Product Delivery Teams to own and operate their products autonomously
  Instill Product Delivery practices in Ops Teams

- **Program:**
  Send every engineer to Hollywood (HQ) for a full week of deep technical training
  21 sessions delivered thus far
  15 unique classes taught over the week

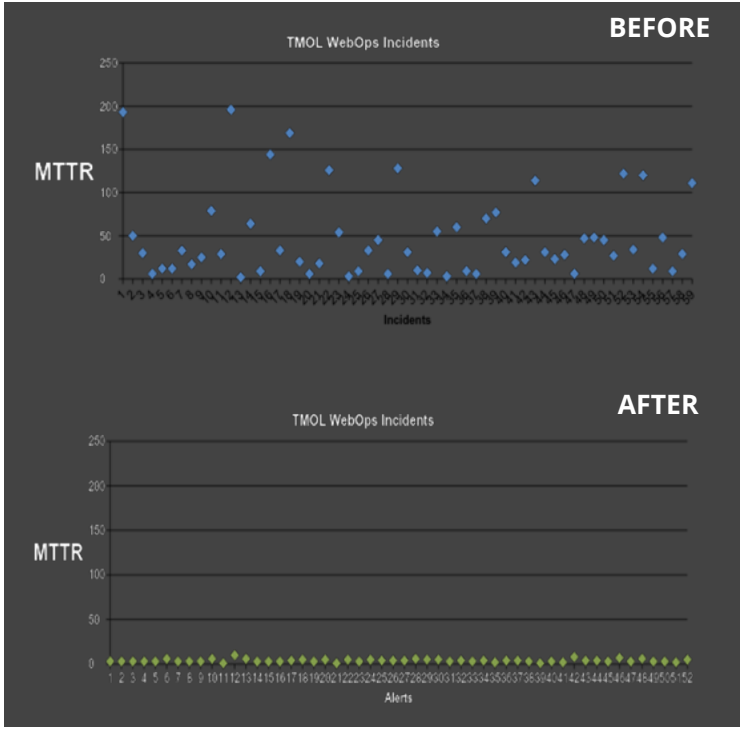
- **Curriculum**:
  System Architecture, Operational Tasks - How to deploy and manage your SW
  Servers, Load-balancers, DNS, Package Installs, Releasing and Scaling
  Software Engineering Practices, TDD, Quality Standards
  Continuous Integration/Delivery
  Docker, Infrastructure-as-Code

[NOW] DEVOPS CROSS-TRAINING

RESULTS:

- Trained ~250 Engineers to be self-sufficient!
- Empowered Teams - Product teams now had the keys to the car AND know how to drive it.
- Significant drop in Friction (waiting for Ops)
- Product teams are now fully On-Call (and loving it!)
- Our “Ownership” culture has blossomed

***Unexpected: Significantly lower MTTR on Incidents (especially large scale outages) – from 47min to 3min**



[THEN] TALKING THE DEVOPS TALK

- Getting faster but not consistently across the board:
  - More deploys, but high degree of variance (inter and intra team)
  - Tech Debt kept growing
  - Quality, Stability and Repeatability wasn't rapidly improving
  - CI/CD wasn't becoming the norm fast enough
  - Muscle Memory - Rebranding old habits/processes as Devops
- Our Devops Transformation ROI was slowing down in spite of leadership drive and support. We needed to:
  - Define what Devops and High-Performance teams look like
  - Provide a clear actionable and measurable roadmap to get there

LESSON:

Define and Measure Performance Objectively

"Quantify the Devops!"

ticketmaster®

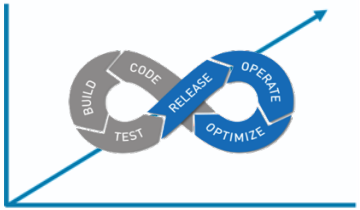
QUANTIFYING OUR MATURITY

- Realized our ability to innovate is dampened by our overly complex software factory:
  - 50-70% of our time spent moving code around ($50M-$90M problem)
  - Over 150 custom-built ways to release products (often manually)
  - Low maturity in key capabilities like monitoring, alerting and testing
  - ~50% of our incidents are caused by issues that should have been detected ***before*** going to production
  - Spending tremendous amounts of effort reinventing wheels

[NOW] WALKING THE DEVOPS WALK

MATURITY MODELS

TECH MATURITY

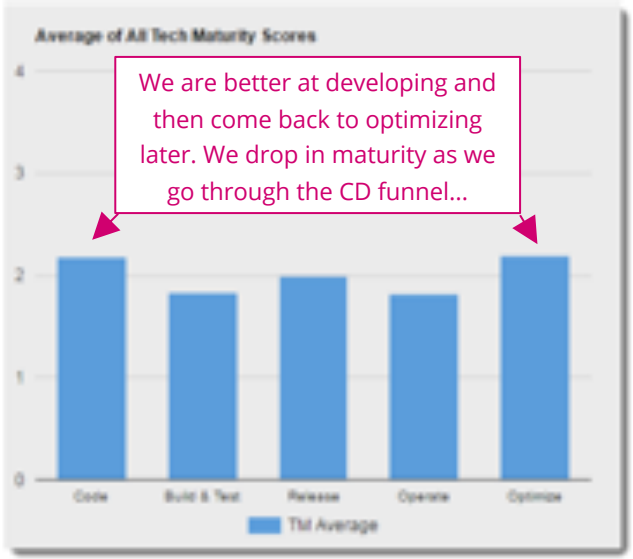


PROCESS MATURITY

TECH MATURITY MODEL

- Developed Tech Maturity model with ~50 capabilities in the dimensions:
  - Code
  - Build & Test
  - Release
  - Operate
  - Optimize
- Every Product team has a clear maturity score
- techmaturity.com - visibility into the state of our products
- Quantify things like Tech Debt!
- Objective scores and clear path for improvement

TECH MATURITY DATA



Capability	Level 1	Level 2	Level 3	Level 4
DevOps Practice	Environments in production are controlled by someone not on the team building the product	Environments in pre-prod are controlled and partially managed by the team building the product and are escalated for issues in that environment	Environments in production are controlled by another team, but owned by the team building the product	Following the DevOps model - i.e. environments in production are fully controlled and owned by the team building the product, including alert/issue escalations

TECH MATURITY MODEL SNIPPET

Category	Capability	Level 1	Level 2	Level 3	Level 4	Min Cloud Readiness
Code	Code Commenting Strategy	Comments are only understood by the person who wrote them	All new code is self-documenting and comments are suitable for documentation generation tools	Most of the code is self-documenting and existing comments are suitable for documentation generation tools	Code comments are consistently suitable for documentation generation tools and code is self-documenting	N/A
	Code Management Strategy	Code is in git, and git is used for release, but there is little to no defined strategy of how to branch/merge/release code	Develop and manage versions from branches	Develop on short-lived (less than 2 weeks) feature branches, and release from merged master	Develop with at least daily code check-ins (using a process allowing traceability to the requested feature) to master and release from master	1
	Test Suite	No/some unit tests, functional tests, critical path tests, performance tests	Some unit tests, functional tests, critical path tests, performance tests with all of them passing successfully	Actively builds and maintains unit tests, functional tests, critical path tests, performance tests (with all of them successfully passing) for positive flows	Actively builds and maintains unit tests, functional tests, critical path tests, performance tests (with all of them successfully passing) for positive and negative flows maintaining 100% critical path coverage	3
	Incremental Coding (prototyping)	No prototyping - e.g. code is not built in a way that a build can be provided to a stakeholder or to a release to test it out prior to completion	Prototypes take effort, but are used to validate/estimate some (i.e. large) features confidently	Prototypes do not take much effort to perform, and are used to validate some features	Code structure is designed in such a way that every feature allows for a quick and iterative prototype - i.e. code can be released and tested at any time	N/A
	Feedback & Requirements	Coding from wireframes, and/or requirements are not fully understood when development is started	Coding from wireframes and comps, and the team understands the requirements and business value of the feature	User feedback with comps are provided before building the feature, and the team has an understanding of how the feature interacts within the ecosystem	User feedback through clickable wireframes and comps are provided before building the feature, and the team has an understanding of how the feature interacts within the ecosystem	N/A
	Logging & Metrics	Default or customized logging and no implementation of metrics	Some metrics and some form of logging	Adherence to established logging and some metrics	Adherence to established logging & metrics standards	2
	Backward / Forward Compatibility	Breaking changes - i.e. tested locally	Changes are regressed by users of the product prior to release	Supports forward compatibility	Coding practices supports backward/forward compatibility	2
	Coding Monitoring & Alerting	Logs have enough data to set up monitoring / alerts on	Some monitoring and some alerting stories are prioritized in the work queue	Prioritization of monitoring & alerting as part of the acceptance criteria	Prioritization of monitoring, alerting, and validation of triggers (e.g. SLAs) as part of the acceptance criteria	2
	Quality Engineering Model	All technology members of the team have separate roles - code or test	Some technology members of the team code and test	Most technology members of the team code and test	Any technology member of the team code and test	N/A
	Code Reuse	Usually codes what they need	Looks to reuse open source or internal code from other projects	Aims to reuse vs rebuild (including internal), and actively evangelises to maximize reuse of products on other teams	Seeks to reuse vs rebuild as part of the inception process (with proven examples), actively evangelises to other teams to maximize code reuse, and	N/A

TECH MATURITY TESTIMONIALS

"The tech maturity has been very, very helpful for us.

...

It helps with organizing our thoughts and area of improvements, which in turn allows us to have an orderly process for improvements.

We have disciplined ourselves and are addressing one or two improvements per iteration."

"We are able to roll out features quicker to production, confidence increased in our ability to accept a specific load."

"We have reduced the instances where we have had to roll back features."

"We familiarized new team members and ensured that ALL team members are now able to support the service."

"If we did not have this process, improvements to our process/service would not be made."

"Tech Maturity has made it easier to have informed discussions around the health of the service and visibility into the usage of the service (who uses it and how?).

Having these dashboards on a TV in the 'Team Space' makes it easier to collaborate on the project with team members."

"After presenting to Product/Business 'how' improving on our Tech debt would make our service more usable and reliable, we were funded for a Reliability investment and a DevOps project.

NOTE: Do not fear justifying Tech Debt Investments to the Business, especially if it is causing harm to our users!"

PROCESS MATURITY MODEL

We *must* deliver value to our fans and clients with shorter turnaround times

- Team maturity around adoption/implementation of agile practices
- Tool to help teams deliver quality products to production faster
- Gives teams insights on areas that need improvement

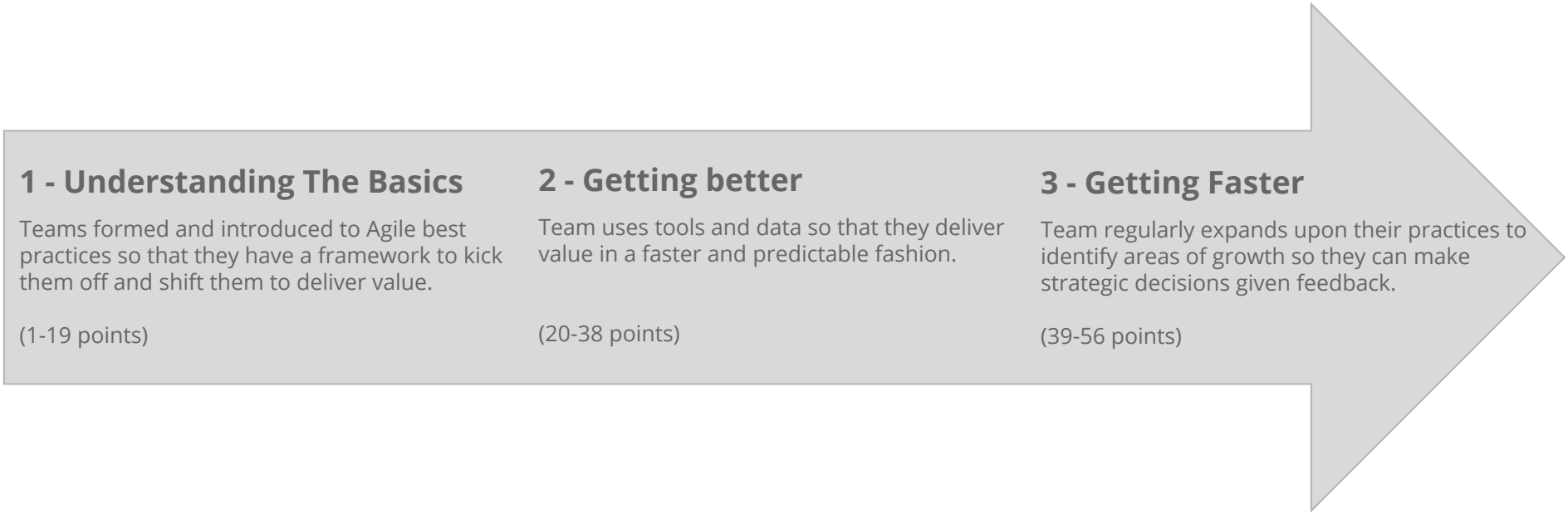
Capability	Level 1	Level 2	Level 3	Level 4
Eliminating Waste	Team has identified columns on their Kanban board and is using the board.	Team has identified WIP (Work in Progress) limits and rules of engagement to move from one column to the next and is following them.	Team tracks cycle time, and is proactively swarming and reducing dependencies and asking for help when there are issues or blockers.	Team proactively is eliminating waste based on cycle time and is streamlining their process.

PROCESS MATURITY MODEL SNIPPET

- Strategic Roadmap
- Inceptions/Workshops
- Roles and Responsibilities
- Measuring Success
- Backlog grooming
- Eliminating Waste
- Pairing
- Thin slicing work
- Prototyping
- Etc

Capability	Level 1	Level 2	Level 3	Level 4
Strategic Roadmap	Team has been exposed to the strategic roadmap and knows how their work aligns to it.	Goal leads are engaged with decisions and tradeoffs for the POV's	Goal, Bet, and POV leadership has been aligned and has a consistent, systematic way to make trade-offs and decisions in real-time	Goal, Bet, and POV leadership has been aligned and has a consistent, systematic and **metric driven way** to make trade-offs (value score for example) and decisions in real-time
Inceptions and workshops	New work is kicked off with an inception, likely the first for the team.	Understands the value of a facilitated workshop and reaches out for facilitiation so that the team can openly collaborate.	Actively collaborating within the team utilizing the facilitation toolkit (hopes and fears, thin slicing, sliders, measures of success, etc.)	Proactively collaborate on each new addition to the backlog and incept or leverage workshops accordingly. 4 in a box team can facilitate or co-facilitate workshops for other teams.
Roles and Responsibilities	Roles and responsibilities have not been defined (who's accountable and responsible for what) but team is working within a "n" in the box structure.	Roles and responsibilities have been identified and the team is working within the goal/bet/pov structure and n in the box.	Roles and responsibilities are clear within your goal and team organization	Roles and responsibilities (4 in a box) are clear outside your goal and team organization.
Mission and Measure of Success	The team has just formed a mission statement and created measures of success. They're not well understood.	The mission statement and measures of success are well understood by the team leads. The team is beginning to track success for POVs.	The mission statement is well understood by the team and the team reports on the measures of success which help inform future direction.	The team has a clear mission statement and references it to ensure new work is aligned. They regularly use the measures of success metrics to validate their slice mission statements.
Backlog	The leads keep a backlog of work.	The backlog of work is prioritized regularly.	The team regularly meets to groom the backlog or the team has an efficient way to groom stories on a regular or as needed basis	The backlog is understood by the team and is aligned to the team and goal missions.
Eliminating Waste	Team has identified columns on their Kanban board and is using the board.	Team has identified WIP (Work in Progress) limits and rules of engagement to move from one column (with exit criteria) to the next and is following them.	Team tracks cycle time, and is proactively swarming and reducing dependencies and asking for help when there are issues or blockers.	Team proactively is eliminating waste based on cycle time and is streamlining their process.
Pairing	The team works individually and occasionally pairs on big issues.	Team is pairing occasionally, but limits themselves not to interrupt the work cycle.	Team is actively trying to pair on most work.	Team has identified and actively uses harmonious team pairings to up-level their team.
Thin Slicing	Team breaks down their work but it often doesn't reflect a slice of functionality.	Team breaks down some of their work into thin slices.	Team breaks down most of their work into thin slices.	Team thin slices regularly and all releases reflect a thin slice of functionality, which they then use to influence the other slices of work.
Prototyping	The team quickly sketches up their idea to visually communicate	The team mocks up a static prototype to visually show the customer an idea of the	The team utilizes a tool such as (invision, api mocks, design jams) to demo concepts to the customer	The team builds a quick usable prototype and gives it to a customer for feedback prior to starting

PROCESS MATURITY SCORING SYSTEM



The true value isn't what level you are at, but the skills you want to acquire next and the steps to get you there.

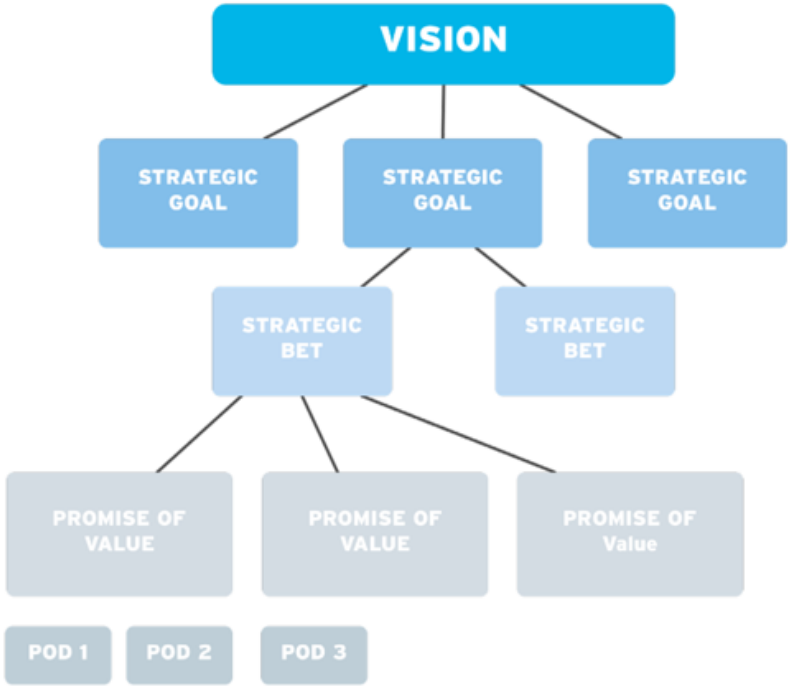
MODELS AND SCORING ARE THE FUTURE

- See huge value in quantifying our maturity
- Scoreboards don’t lie
- We are currently building and tuning:
  - Risk Assessment Model
  - Technology (Cost) Optimization Model
  - Model Optimization Model (Just kidding..)

** We are Open-Sourcing them soon. Ping me or Alex Hazboun (Chief Architect) to get involved

[THEN] LEAN TRANSFORMATION

- Create Value Alignment
- Managing our company like an investment portfolio
- Portfolio organized by Goals / Bets / Promises of Value (POV)
- Every team aligned with a POV is an active investment



FACEBOOK “WALL” - We are doing it wrong

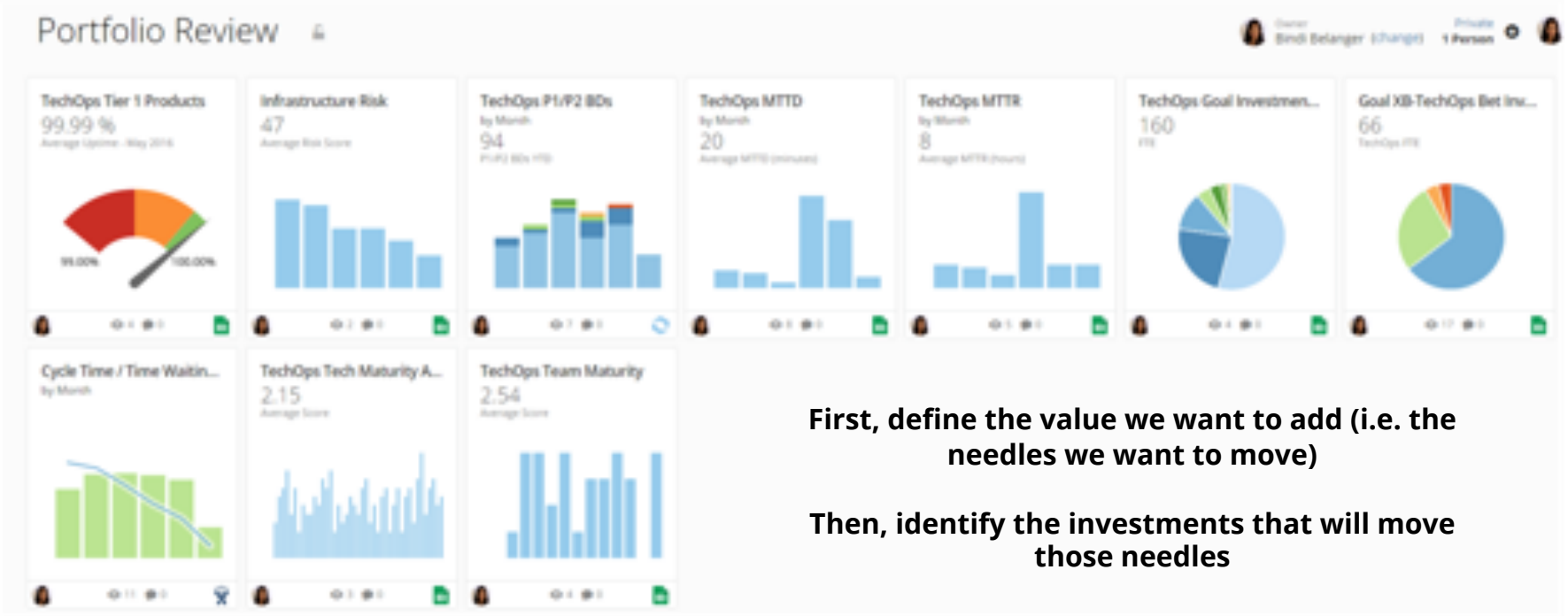
- New structure but we brought all of our old processes
- “Value Tagging” but not measuring actual value delivery
- We built a “WIP” machine

LESSON:

Outcome > Output

ticketmaster®

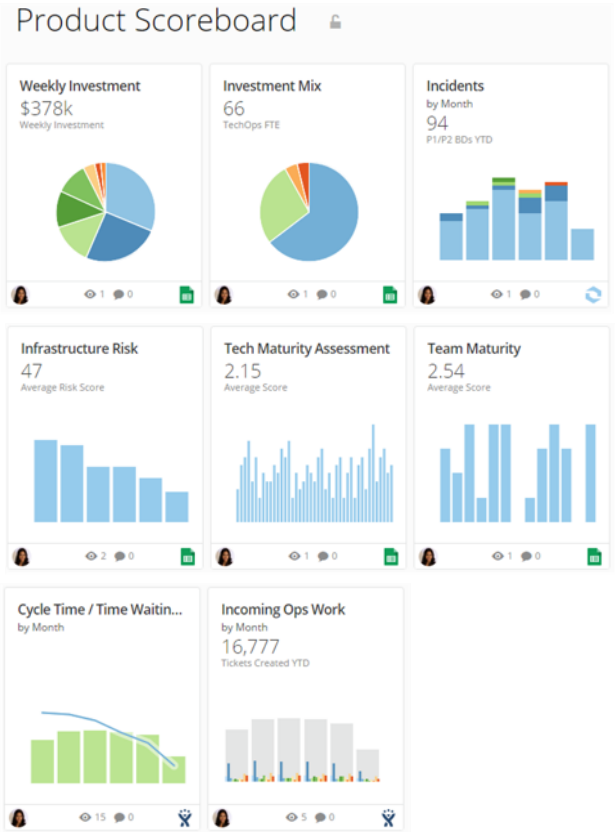
[NOW] VALUE DRIVEN DELIVERY



First, define the value we want to add (i.e. the needles we want to move)

Then, identify the investments that will move those needles

[NOW] SCOREBOARDS



LEADERS:

- Ability to make decisions on investments based on which “needle” they want to move
- Ability to understand the impacts of investments

TEAMS:

- Every product team has a scoreboard
- Clear understanding if we’re winning or not

[NOW] LEADERSHIP CULTURE

- Get comfortable iterating on your Org - Performance/Value created > Org size
- “People don’t create value up to their boss, they create value towards the customer” - Jody Mulkey
- Dashboards > Words = “Move-the-Needle” culture
- Total Transparency - Product Inventory, Domo dashboards, Maturity scores, Product Cost, Team and Product performance
- Visibility = Leaders are truly becoming CEO’s of their Products (Business)

Continuing our Journey...

- We now have Autonomous Product Teams! - Can build new environments in minutes vs weeks/months.
- Now we have the structure, visibility and clear roadmap to high performance.
- We are heading full steam ahead leveling up our teams and products for the future.

ticketmaster®

Contact me:
justin.dean@ticketmaster.com

ticketmaster®