Visual Studio Online: Journey to DevOps

Thoughts to share from Microsoft Developer Division's transformation to DevOps

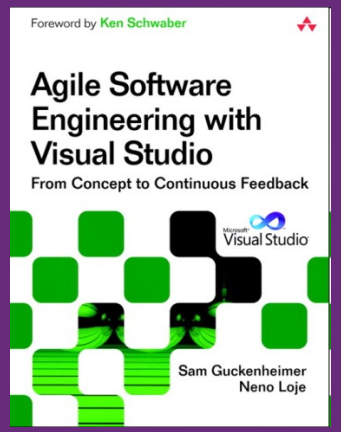
Sam Guckenheimer

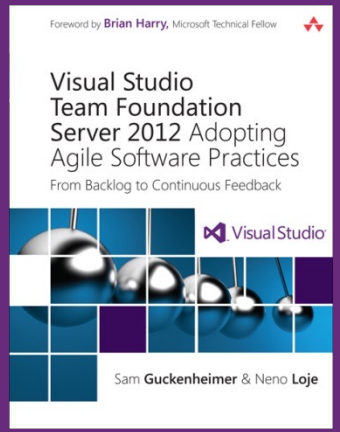
Product Owner, Visual Studio Product Line
Microsoft Corporation

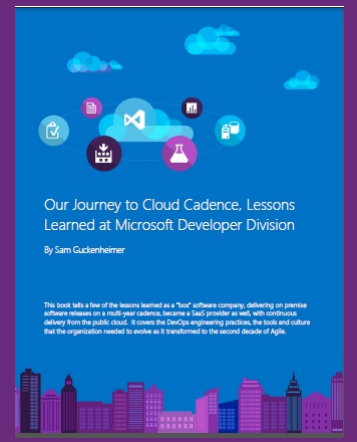
@samguckenheimer

@shitsamgusays

http://aka.ms/devops



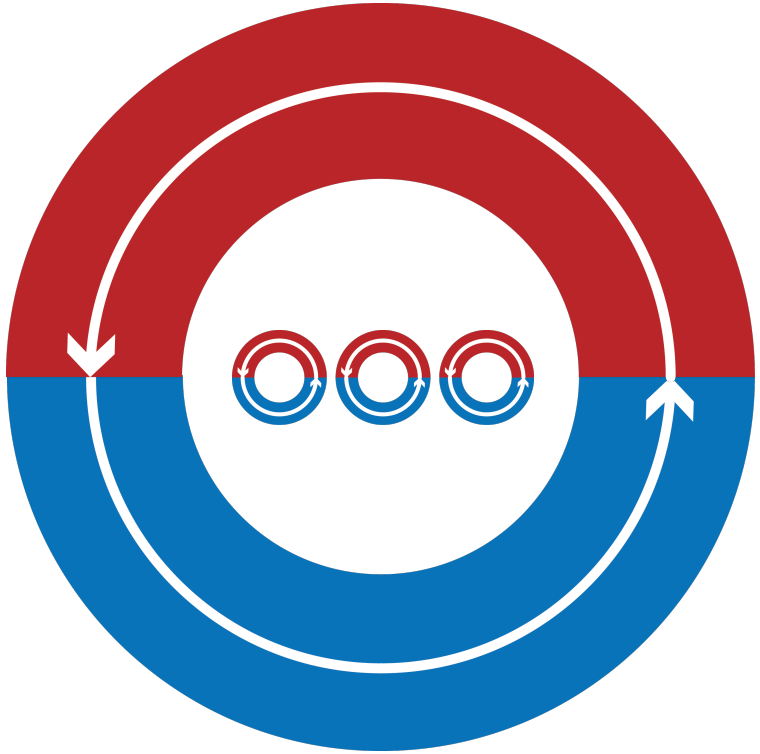




Visual Studio

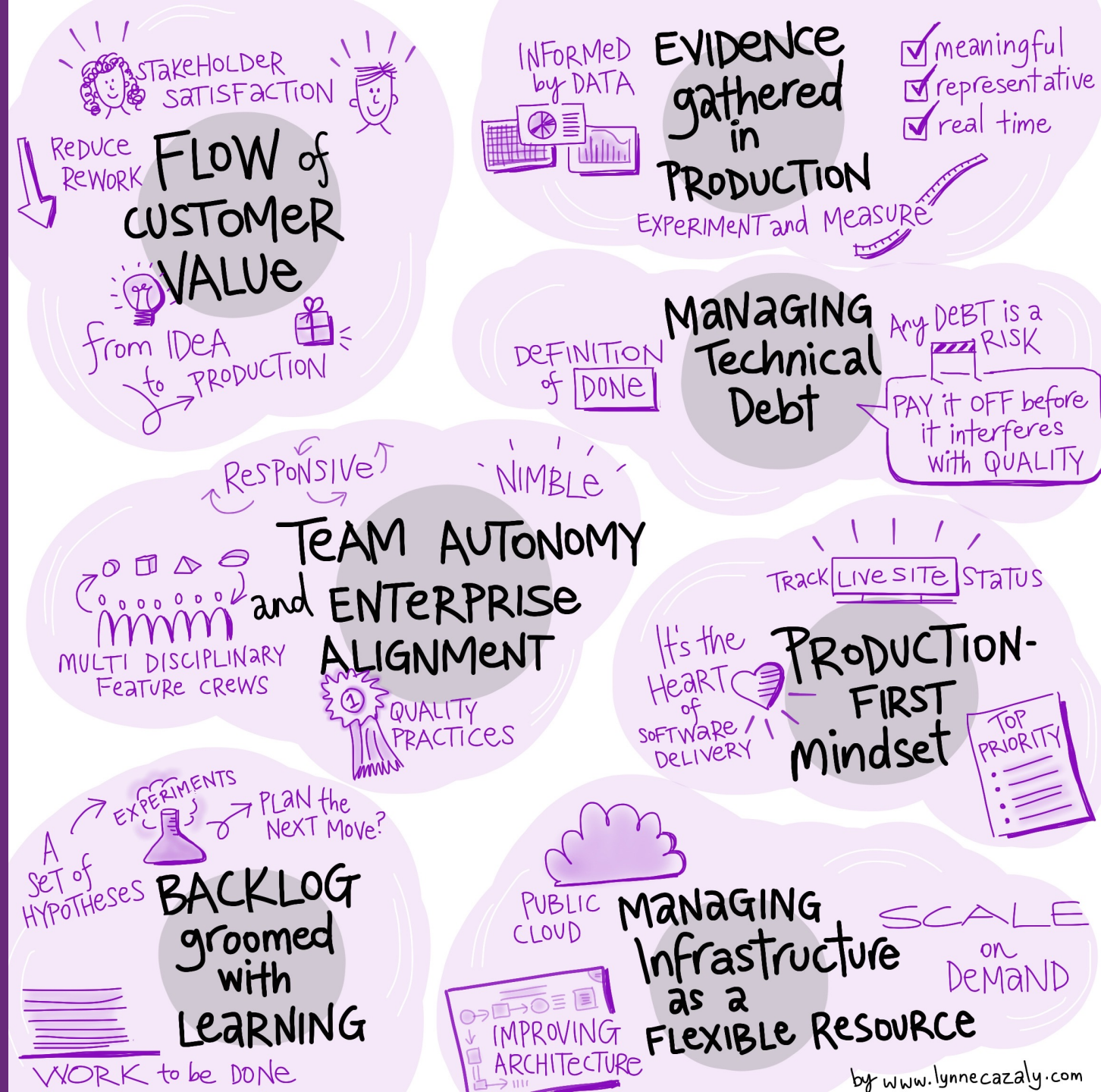
Canonical view of DevOps (aka the "Three Ways")

Development
Business

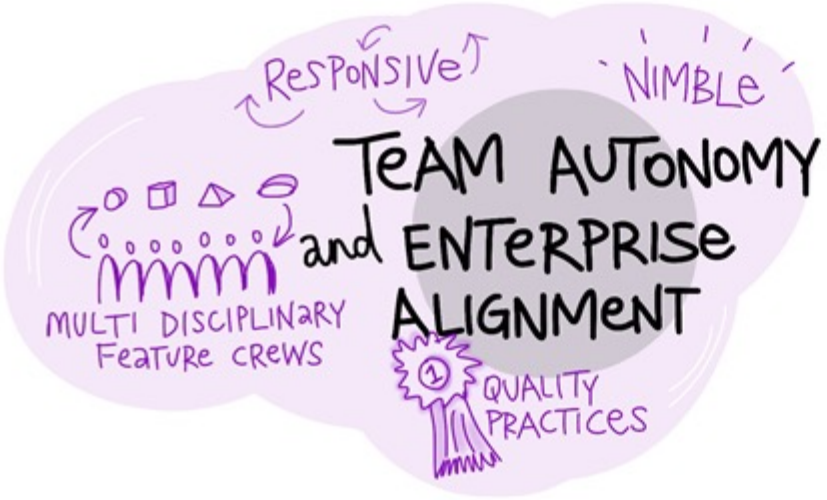


Operations
Customer

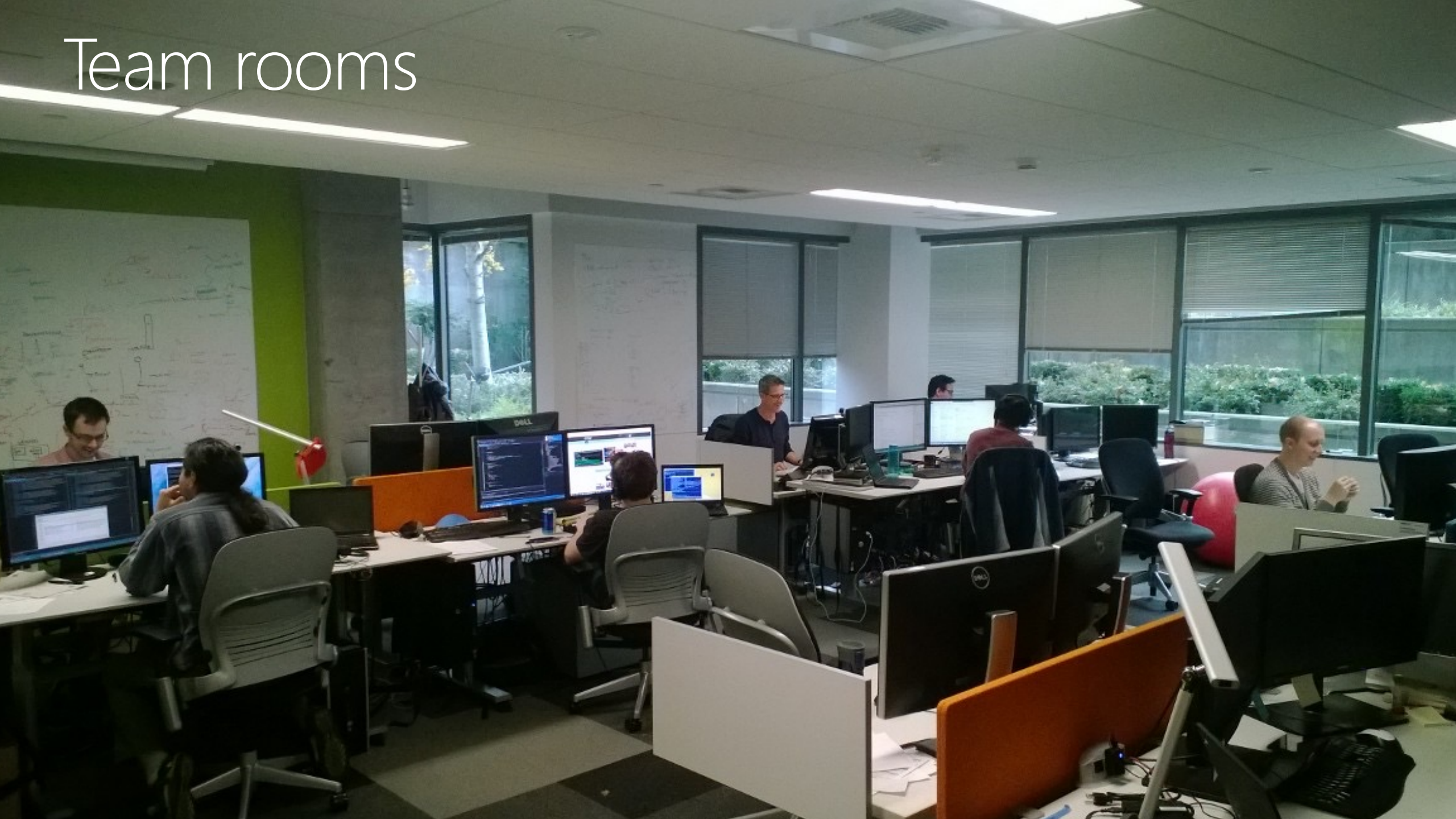
Our learnings as a SaaS provider



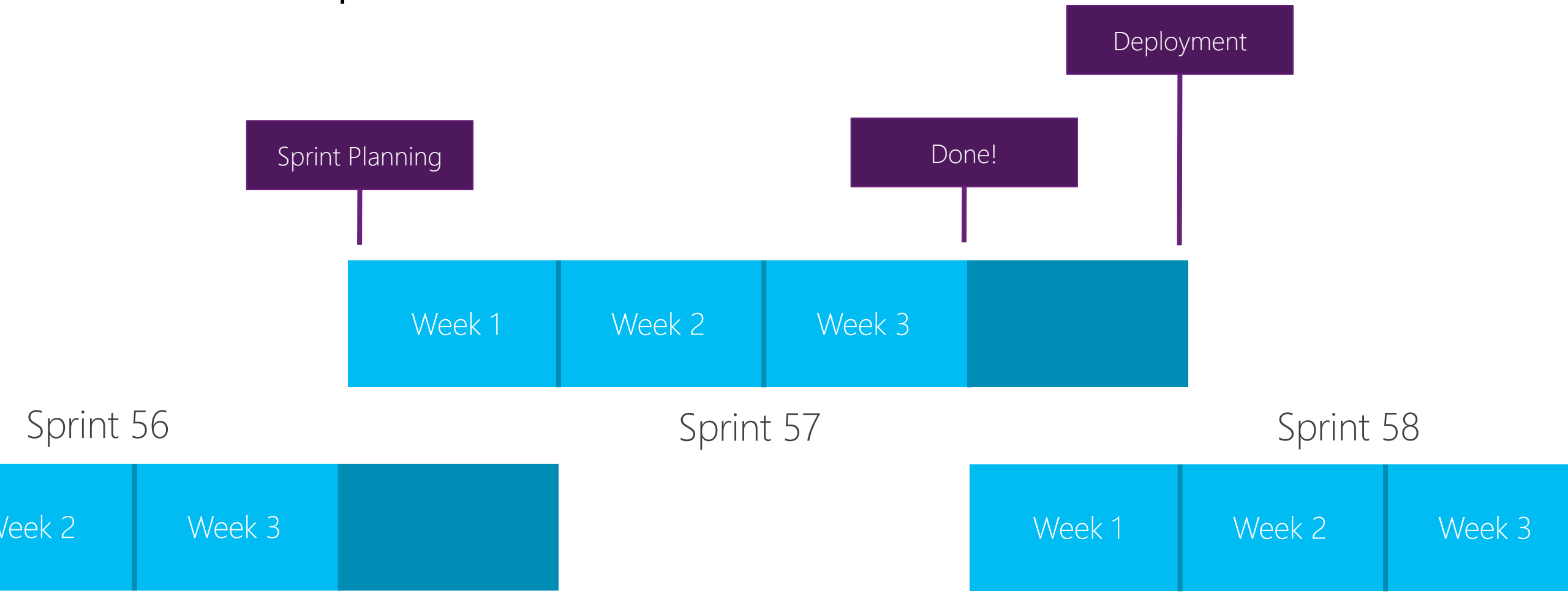
Our learnings as a SaaS provider



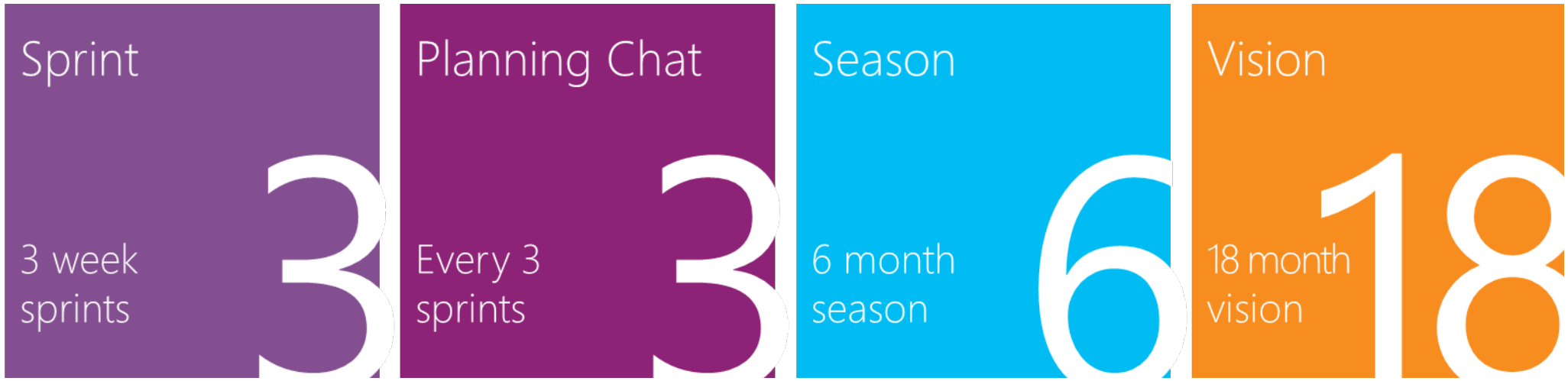
Team rooms



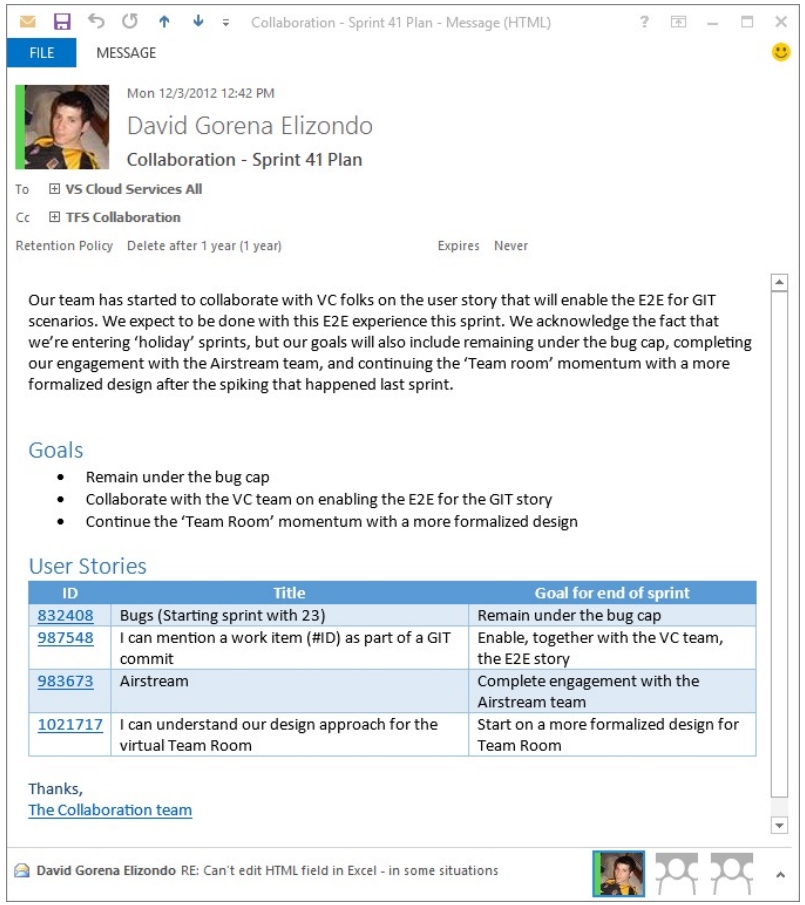
3 week sprints



Planning horizons



Sprint mails



Collaboration - Sprint 41 Plan - Message (HTML)

FILE MESSAGE

Mon 12/3/2012 12:42 PM

David Gorena Elizondo

Collaboration - Sprint 41 Plan

To ⊞ **VS Cloud Services All**

Cc ⊞ **TFS Collaboration**

Retention Policy Delete after 1 year (1 year) Expires Never

Our team has started to collaborate with VC folks on the user story that will enable the E2E for GIT scenarios. We expect to be done with this E2E experience this sprint. We acknowledge the fact that we're entering 'holiday' sprints, but our goals will also include remaining under the bug cap, completing our engagement with the Airstream team, and continuing the 'Team room' momentum with a more formalized design after the spiking that happened last sprint.

Goals

- Remain under the bug cap
- Collaborate with the VC team on enabling the E2E for the GIT story
- Continue the 'Team Room' momentum with a more formalized design

User Stories

ID	Title	Goal for end of sprint
832408	Bugs (Starting sprint with 23)	Remain under the bug cap
987548	I can mention a work item (#ID) as part of a GIT commit	Enable, together with the VC team, the E2E story
983673	Airstream	Complete engagement with the Airstream team
1021717	I can understand our design approach for the virtual Team Room	Start on a more formalized design for Team Room

Thanks,
The Collaboration team

David Gorena Elizondo RE: Can't edit HTML field in Excel - in some situations

Collaboration - Sprint 41 Results - Message (HTML)

FILE MESSAGE

Thu 1/3/2013 9:38 AM

Justin Marks

Collaboration - Sprint 41 Results

To ⊞ **VS Cloud Services All**

Cc ⊞ **TFS Collaboration**

You forwarded this message on 1/7/2013 8:54 AM.

Even with the holidays the team was able to get a bunch of stories completed. The Collaboration team has really embraced collaborating with other feature teams:

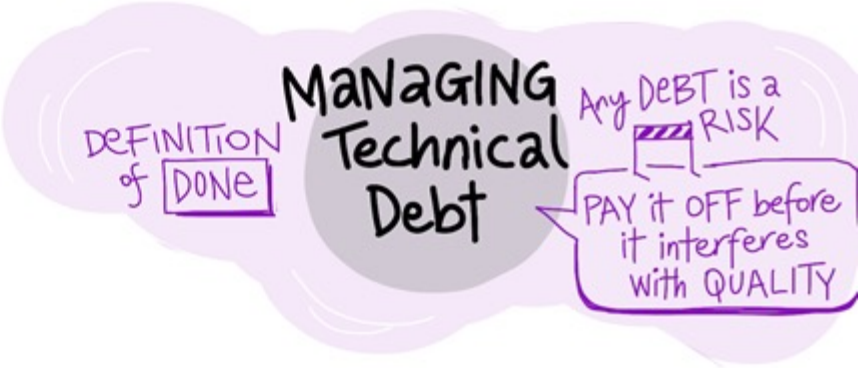
- We've had a strong collaboration with the VC team on delivering an E2E story around linking GIT commits to work items as can be seen in the below sprint video
- We worked with the Airstream team to pull down and validate the RI so that the discussion service is now being built in production using Airstream
- With the help of the WIX team we added the ability to send email from the product backlog

All the while we continued to stay on top of bugs dropping below 20!

Sprint Demo

Justin Marks RE: TFS 2012 - Excel integration: Hide "hidden" work item types

Our learnings as a SaaS provider

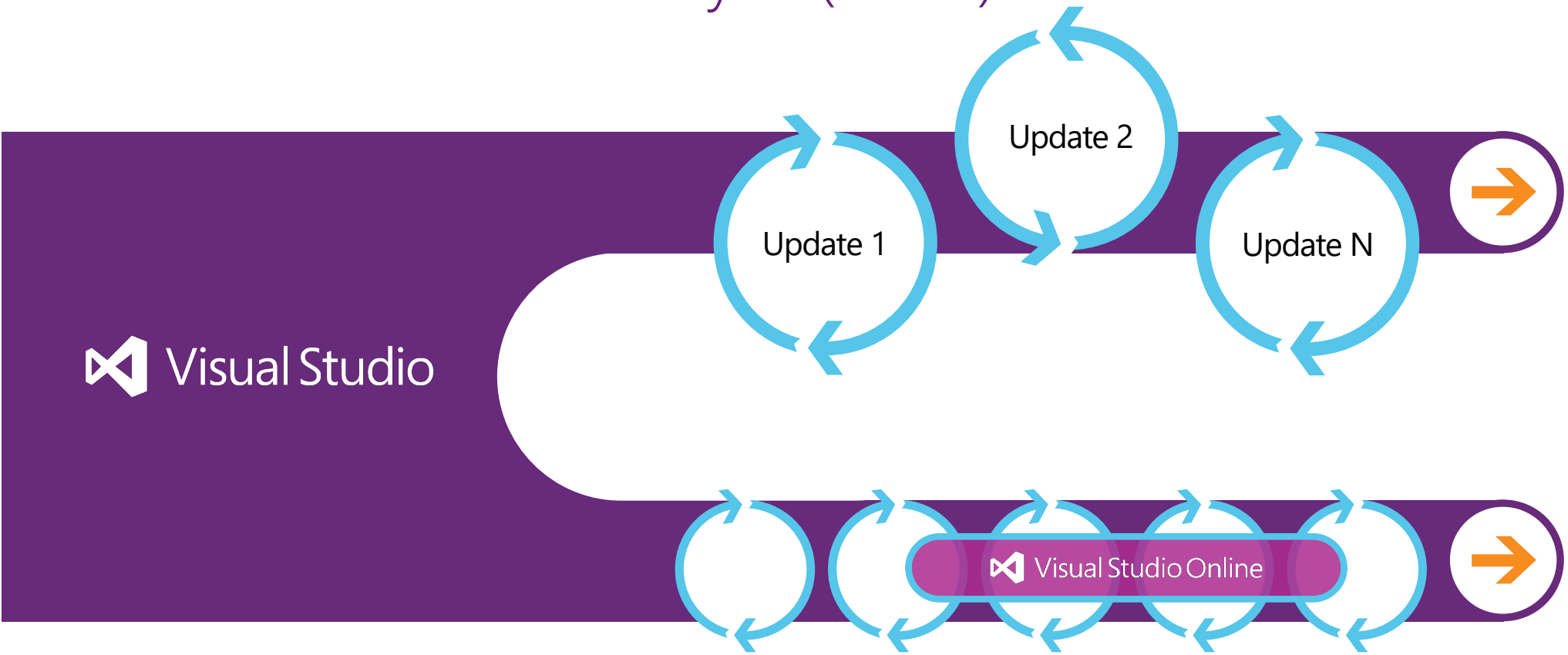


Code: Cloud first, then move on-premises

One code base with multiple delivery streams

Single master branch, multiple release branches

Shared abstraction layer (VSSF)



Goals: Stay clean and move fast

Ship frequently with minimum friction

Dream of a system that can do hands-free push of a hotfix to production in 15 minutes

Fast and predictable inner loop

Inner loop = Build>Deploy>Test cycle which runs reliably hundreds of times a day

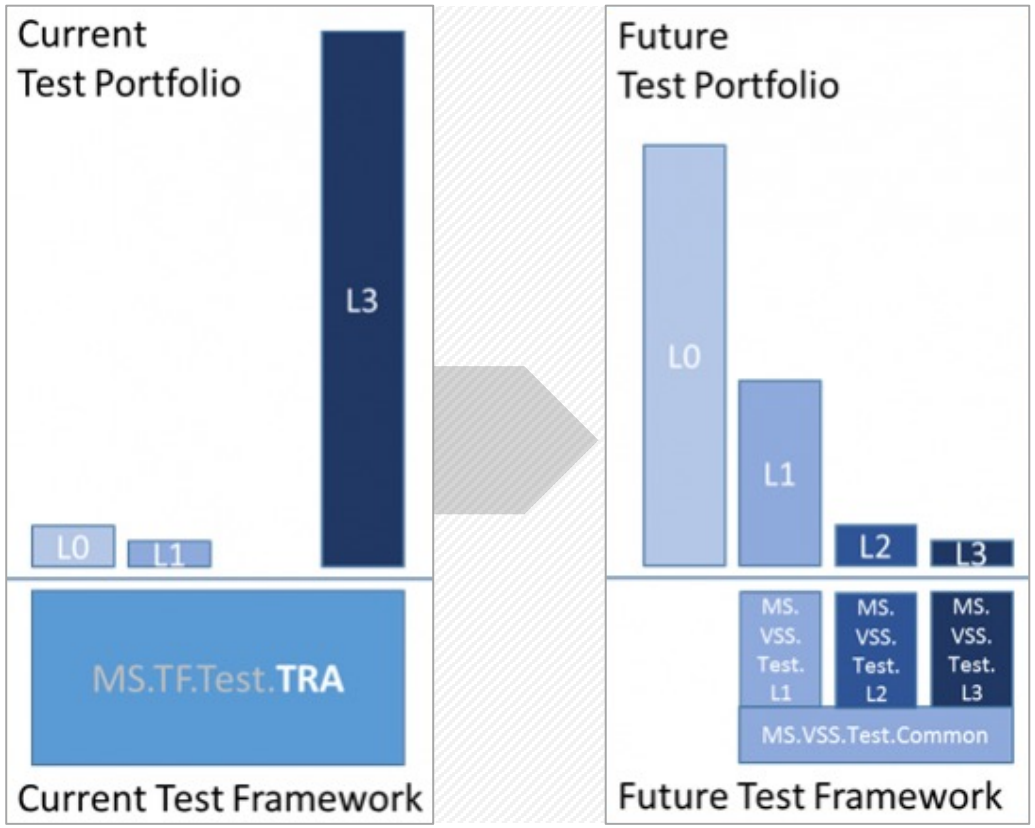
Quality approach designed for Cloud cadence

Build confidence through staging, exposure control & rich telemetry

Continue to meet a rigorous on-premises quality bar

Same definition of done must work for all delivery streams

Test at the lowest possible level



Principles

Tests should be written at the lowest level possible

Write once, run anywhere - includes production system

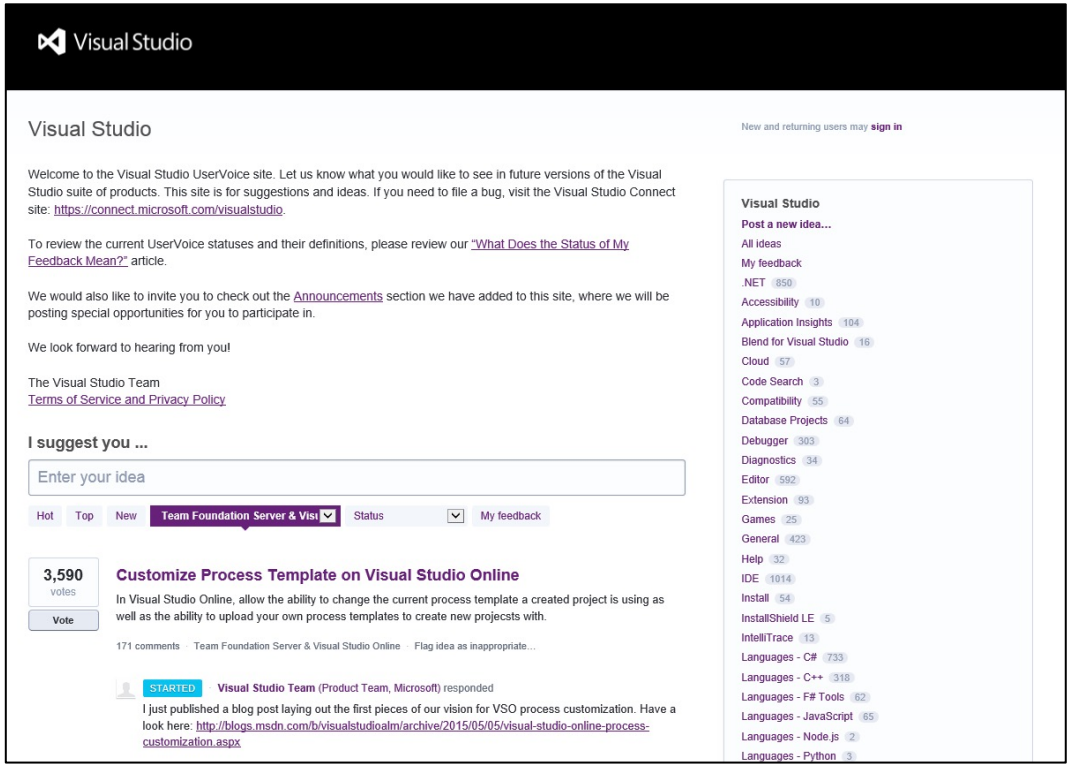
Product is designed for testability

Test code is product code, only reliable tests survive

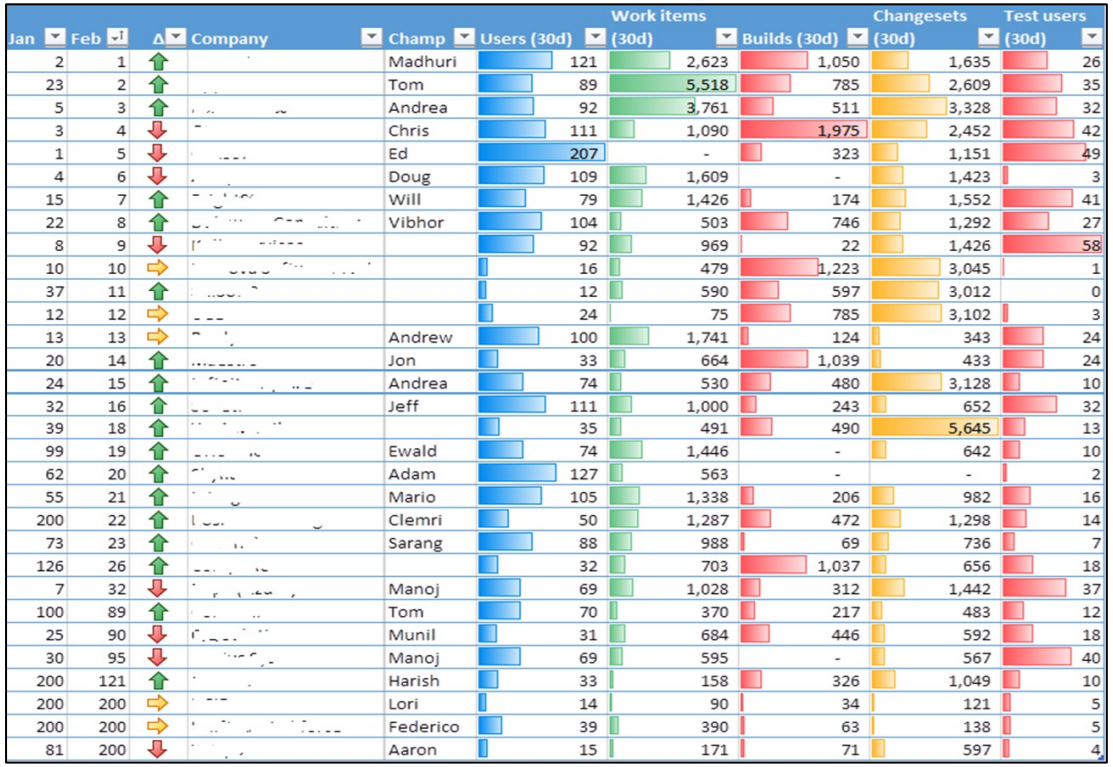
Testing infrastructure is a shared service

Our learnings as a SaaS provider

Talk to customers



UserVoice



Jan	Feb	Δ	Company	Champ	Users (30d)	Work items (30d)	Builds (30d)	Changesets (30d)	Test users (30d)
2	1	↑		Madhuri	121	2,623	1,050	1,635	26
23	2	↑		Tom	89	5,518	785	2,609	35
5	3	↑		Andrea	92	3,761	511	3,328	32
3	4	↓		Chris	111	1,090	1,975	2,452	42
1	5	↓		Ed	207	-	323	1,151	49
4	6	↓		Doug	109	1,609	-	1,423	3
15	7	↑		Will	79	1,426	174	1,552	41
22	8	↑		Vibhor	104	503	746	1,292	27
8	9	↓			92	969	22	1,426	58
10	10	→			16	479	1,223	3,045	1
37	11	↑			12	590	597	3,012	0
12	12	→			24	75	785	3,102	3
13	13	→		Andrew	100	1,741	124	343	24
20	14	↑		Jon	33	664	1,039	433	24
24	15	↑		Andrea	74	530	480	3,128	10
32	16	↑		Jeff	111	1,000	243	652	32
39	18	↑			35	491	490	5,645	13
99	19	↑		Ewald	74	1,446	-	642	10
62	20	↑		Adam	127	563	-	-	2
55	21	↑		Mario	105	1,338	206	982	16
200	22	↑		Clemri	50	1,287	472	1,298	14
73	23	↑		Sarang	88	988	69	736	7
126	26	↑			32	703	1,037	656	18
7	32	↓		Manoj	69	1,028	312	1,442	37
100	89	↑		Tom	70	370	217	483	12
25	90	↓		Munil	31	684	446	592	18
30	95	↓		Manoj	69	595	-	567	40
200	121	↑		Harish	33	158	326	1,049	10
200	200	→		Lori	14	90	34	121	5
200	200	→		Federico	39	390	63	138	5
81	200	↓		Aaron	15	171	71	597	4

Top Customers

Matching SLA to user experience

Phase 1: Outside-in synthetic tests

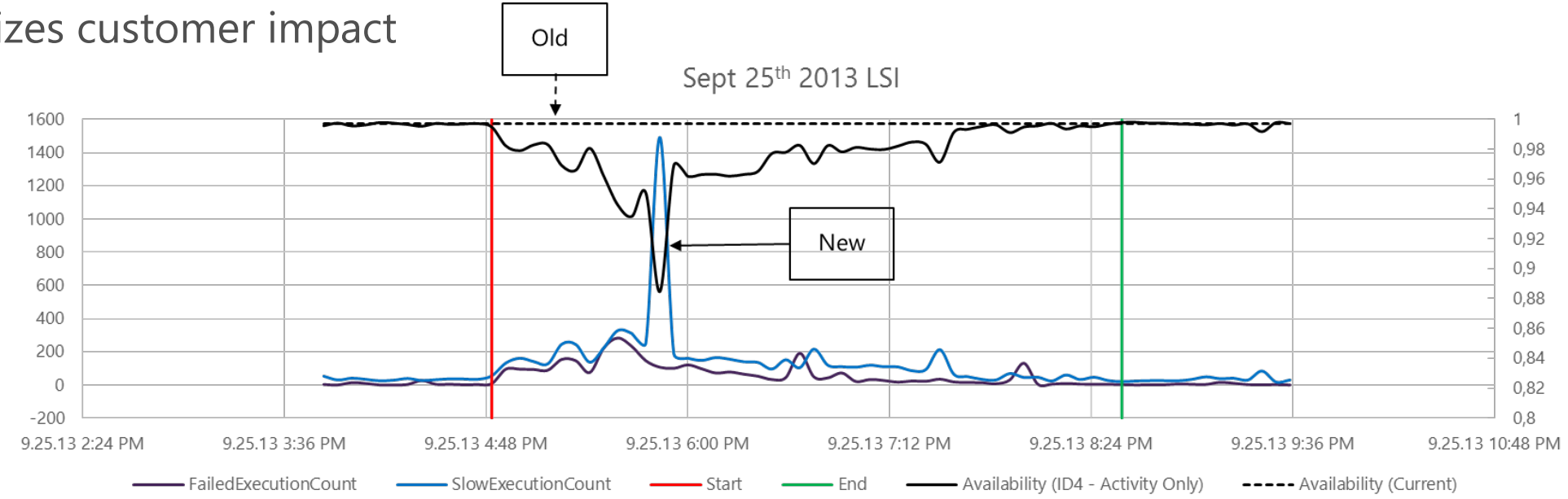
Experience: Coverage too narrow as service footprint grows

Phase 2: Command health

Experience: Loses sensitivity as command volumes grow

Phase 3: Failed or slow user minutes per account

Experience: Empathizes customer impact



Focus on the outliers (“Embrace the Red”)

We measure availability by account …

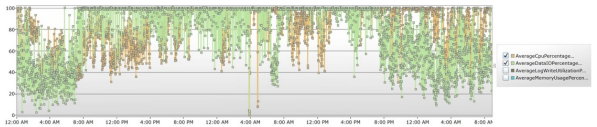
… and proactively reach out to customers with low availability

From: Ladislau Szomoru
Sent: Thursday, May 21, 2015 2:42 AM
To: Madhu Kavikondala; Harish Thekethil; Chandru Ramakrishnan; Ed Glas; Buck Hodges; Munil Shah
Subject: OneDrive.visualStudio.com

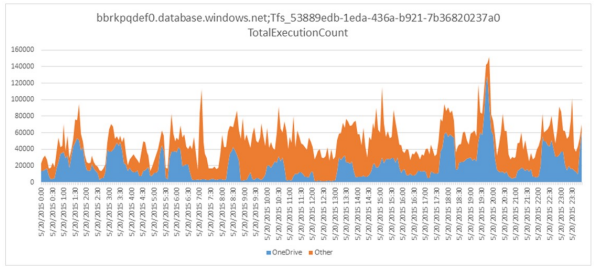
Hello All,

I am currently working on a stateful AIMS rule to fill a monitoring gap that we have around SQL Azure resource stats. I have already deployed the rule for testing purposes (alerts are routed to me only) and I am currently doing the analysis of the generated alerts in order to tweak the settings of the rule, as well as flush out any product bugs before I start routing the alert to VSOLS.

As part of this exercise I came across database bbrkpqdef0.database.windows.net;Tfs_53889edb-1eda-436a-b921-7b36820237a0 which is running very hot in CPU. Taking a closer look at the database we noticed that OneDrive.visualstudio.com is hosted in this database. We have already filed a product bug (323883) as prc_DeleteUnusedFiles that was running for hours on this account causing 100% CPU, but even after we have disabled the cleanup job, the database still spikes up to 100% CPU.



Looking at the telemetry from 2015-05-20, I see that the OneDrive account represents 40% of the total commands in the problematic partition database. Below I have included a stacked area chart that breaks down the total commands between OneDrive and all the other accounts hosted in the partition database:



Based on the telemetry, I would like to do the following:

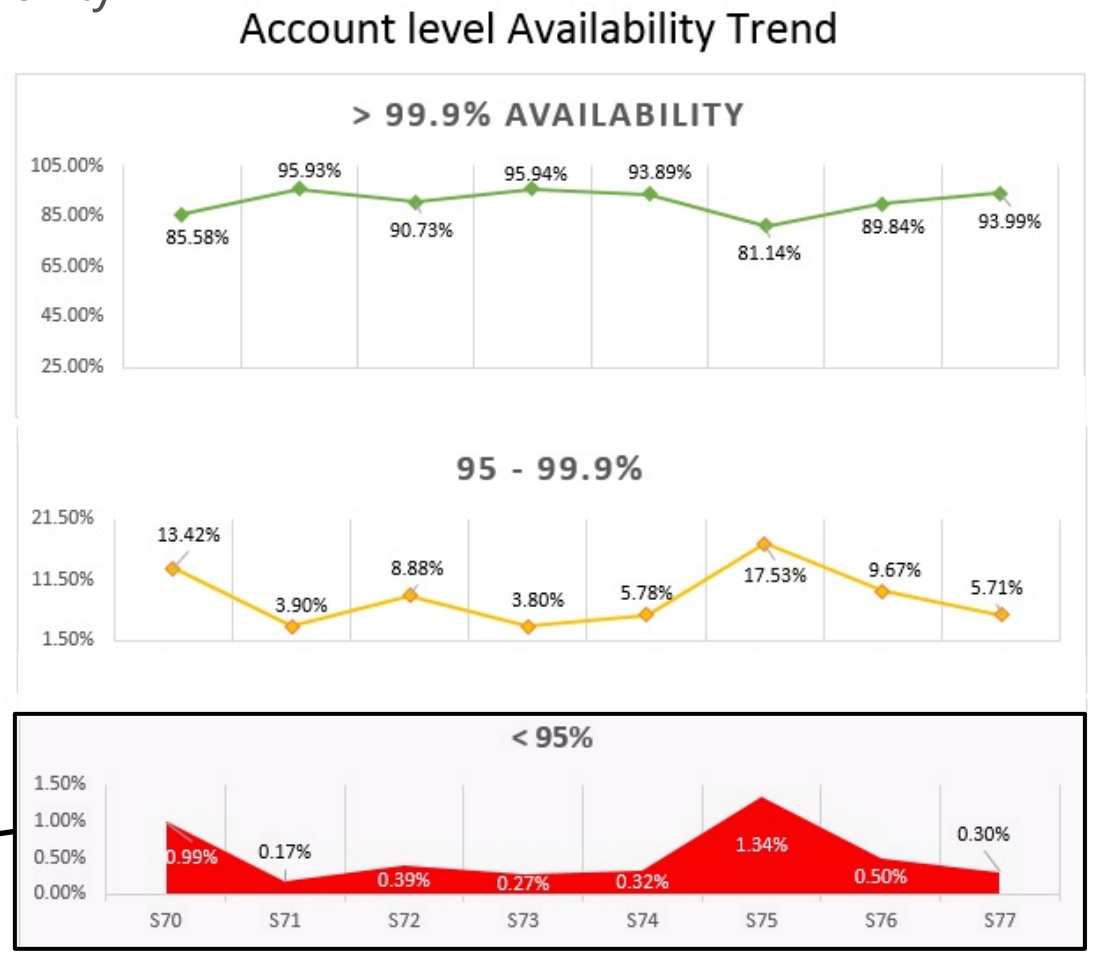
1. The problematic partition database is currently a P2, and I would to ask for your permission to increase it to P3 in order to provide immediately relief to the other customers hosted in the same partition database. After the database is increased to P3, I will continue to look at the telemetry and peel the onion to make sure that any product bugs are being addressed.
2. Given the heavy usage of the OneDrive account, I believe that we should kick off the conversation with ASG about the long term plan of this account. If the account is here to stay, I would like to suggest that we move it to TFS SU2 in a single-tenant partition database hosted on the data tier that was purchased to ASG. Thoughts?

Let me know if you have any questions or concern regarding the telemetry. Thanks!

Ladislau

End of document

Found one of the top customers with low availability. Proactively reached out and resolved their issue.



We took inspiration from …

Open Source Software (OSS) …

OSS Workflows are all about sharing, reuse & collaboration

Componentization for sharing & collaboration are self re-enforcing

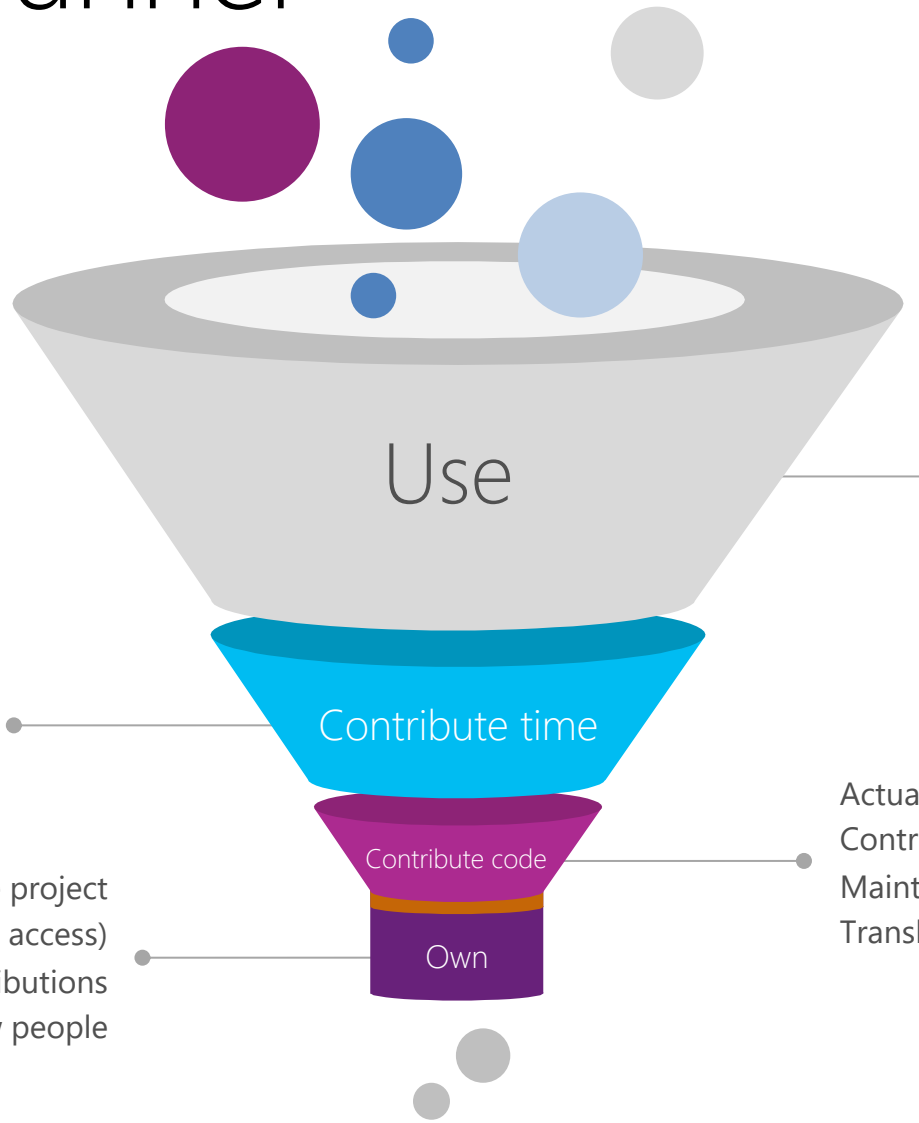
… and large-scale services …

Loosely coupled, collaborating services that ship independently

… with Enterprise rigor

Enterprise code governance and requirements for products supported over a long period

Contribution funnel



Be able to understand what it does
Can easily pick it up and use
Fork/Follow/Favorite
Download

Answer questions, write blog posts
Fix/add documentation
Log bugs
Fix typos

Actually contribute code patches that fix bugs
Contribute entirely new features
Maintain platforms
Translate

Stick around - influence the direction of the project
Become a core committer (get write access)
Accept/validate code contributions
Nurture new people

Our learnings as a SaaS provider

Build-Measure-Learn

Hypothesis

We believe {customer segment} wants {product/feature} because {value prop}

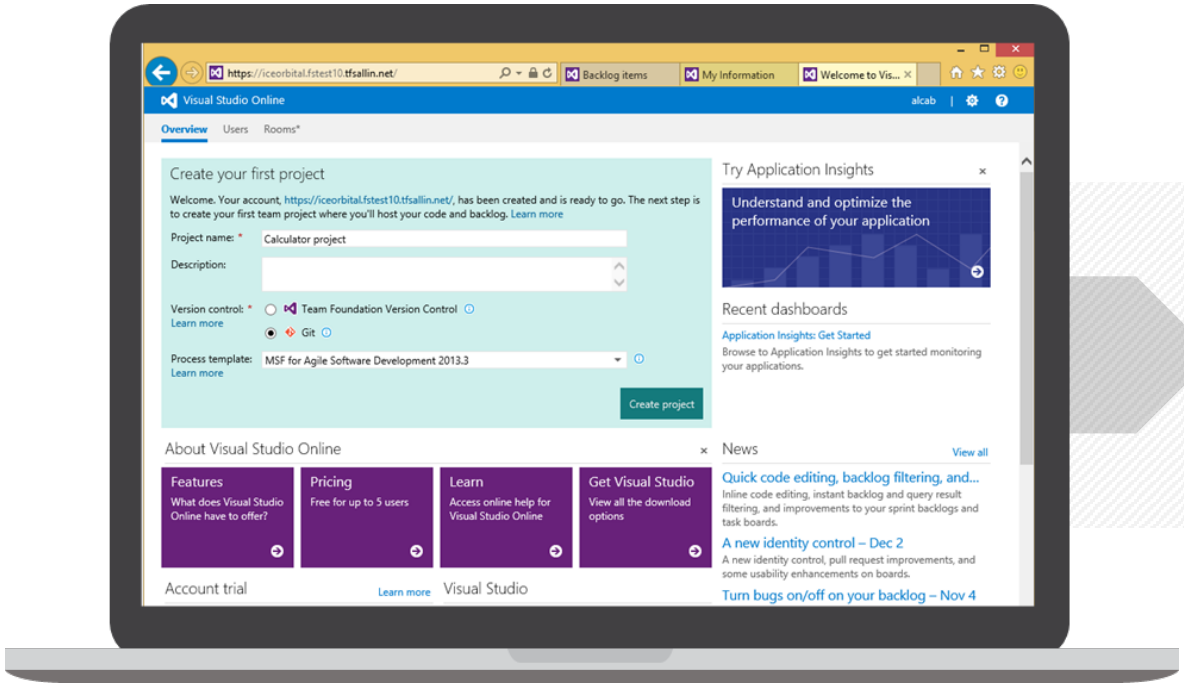
Experiment

To prove or disprove the above, the team will conduct the following experiment(s): …

Learning

The above experiment(s) prove(s) the hypothesis by impacting the following metric(s): …

Before …

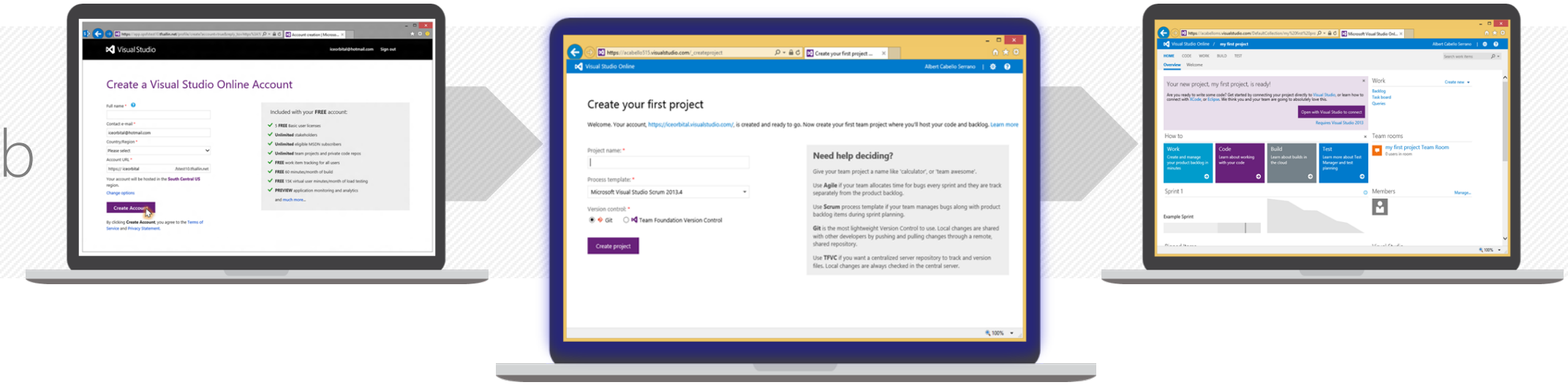


Previous project creation experience

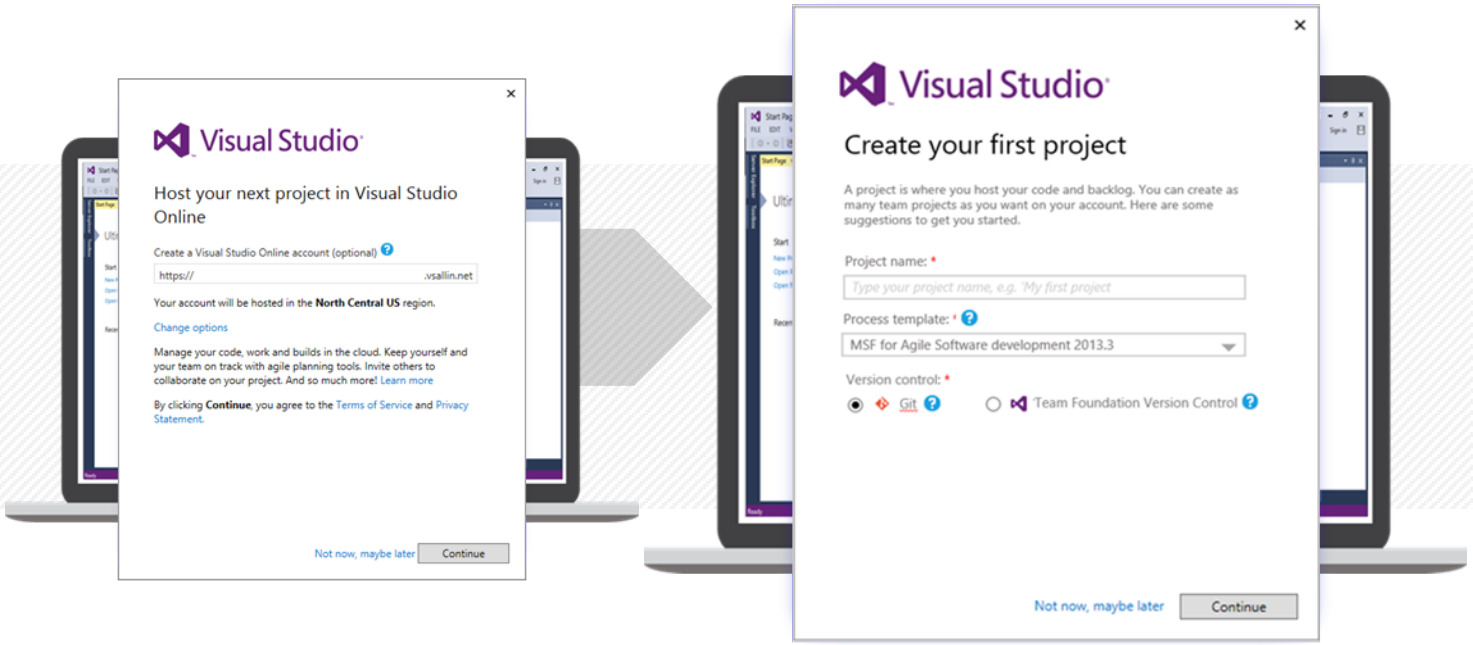
Too many actions and distractions can prevent customers from taking the next natural step of creating a new project

After …

Web



IDE

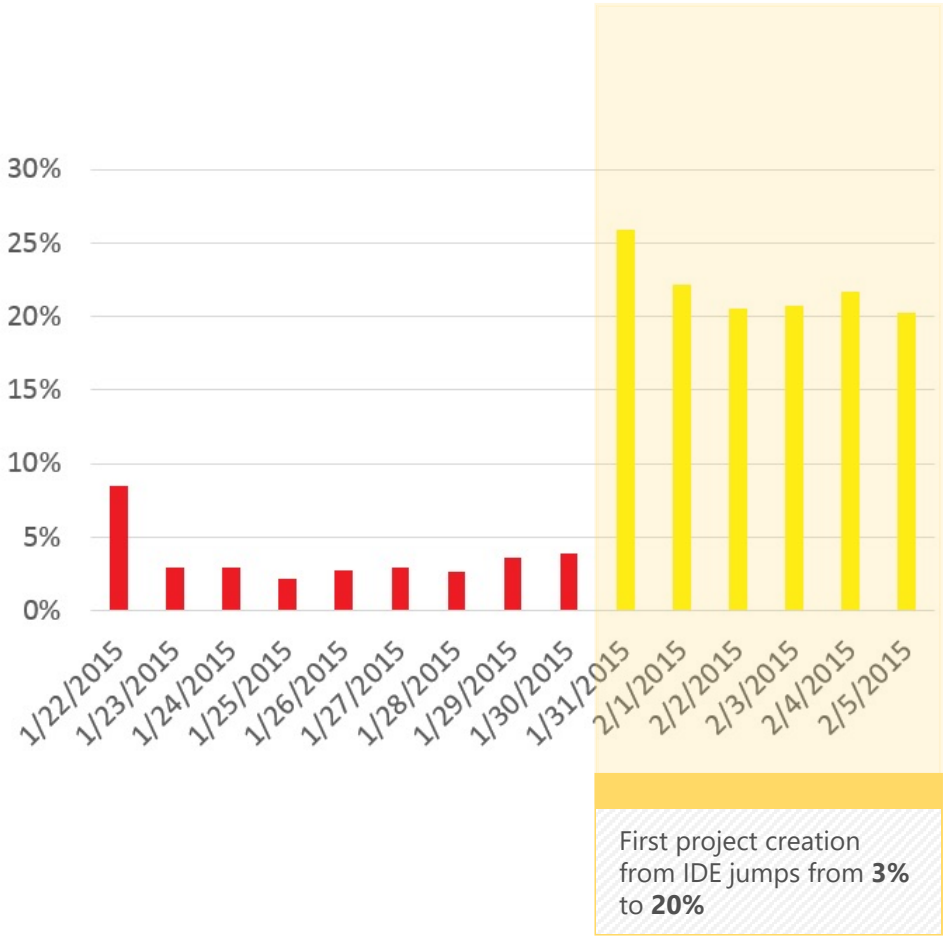


Focused project creation experience – integrated into the IDE flow

Validated learning

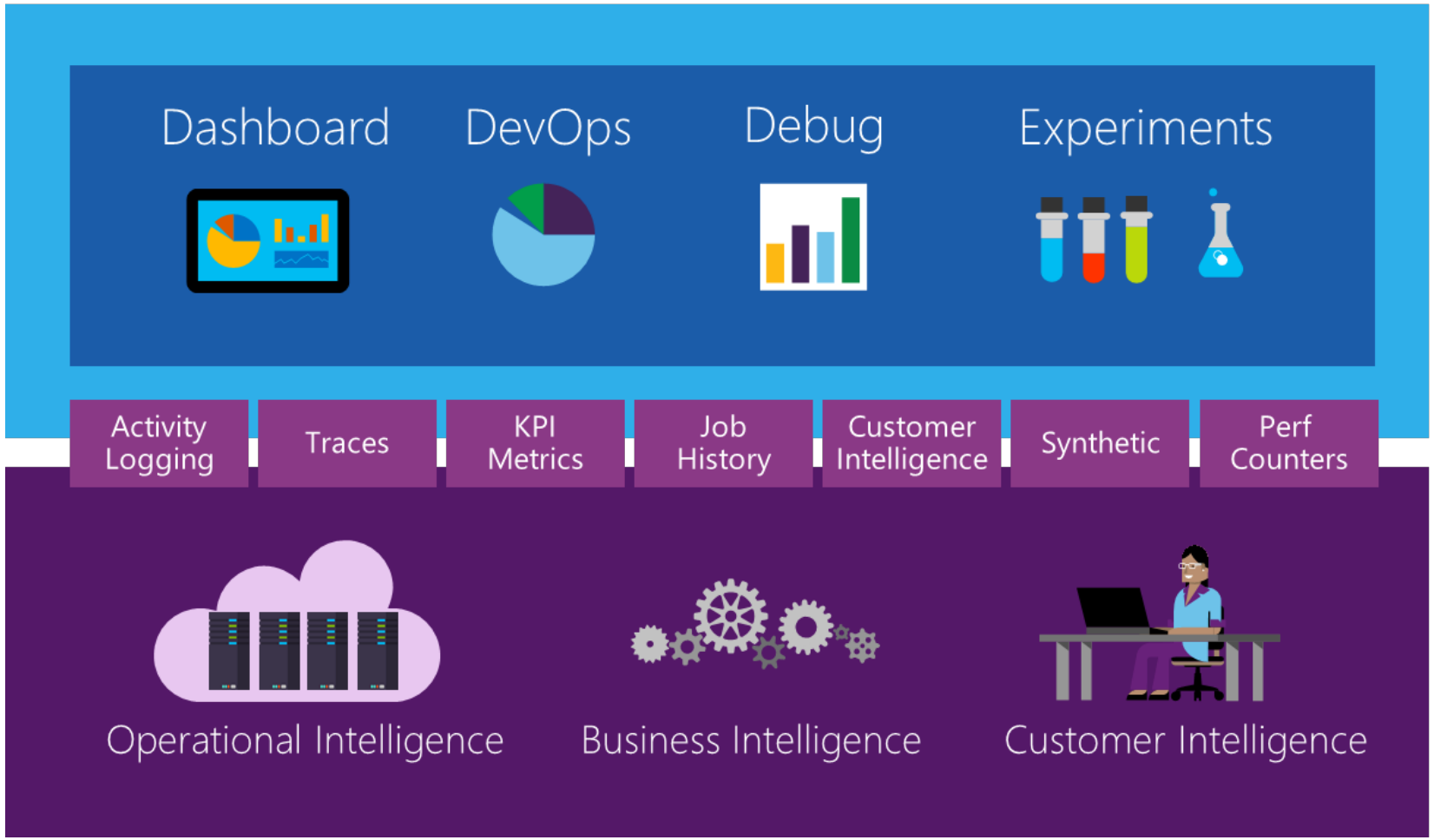
% of accounts creating projects the day of account creation
January 22 to February 11 (all sources)

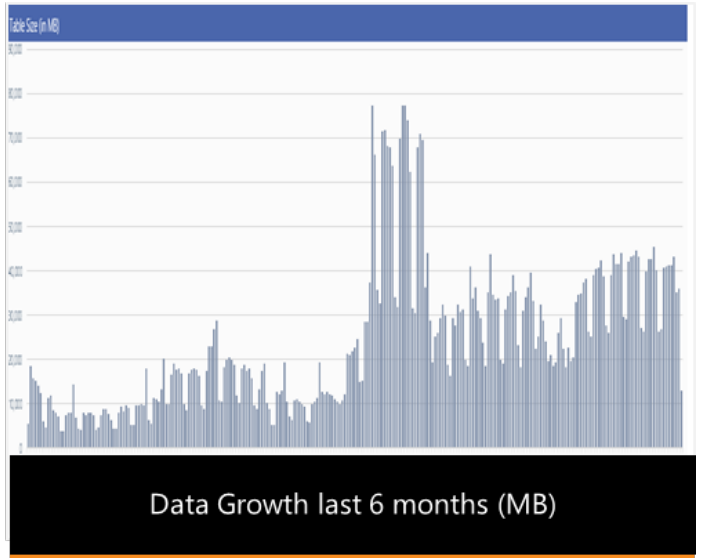
% of accounts creating projects the day of account creation
January 22 to February 5 (IDE only)



Our learnings as a SaaS provider

We collect everything!





Data Growth last 6 months (MB)

Volume

~60GB

Average per day

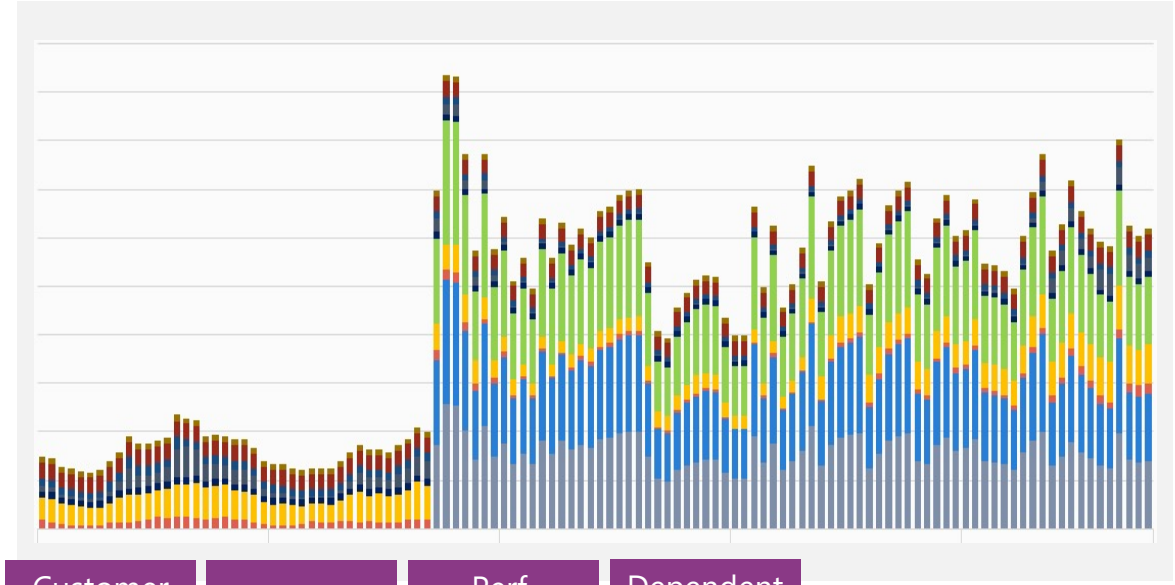
Traces

Traces states in the code, error conditions

Lots of knobs (user, component, statement), all errors on by default

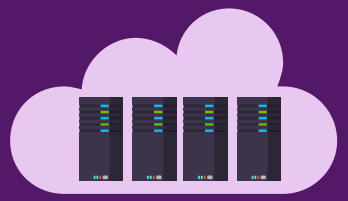
Great for debugging live site incidents

Up to 150gb per day



Visual Studio Online

Activity Logging | **Traces** | KPI Metrics | Job History | Customer Intelligence | Synthetic | Perf Counters | Dependent Services



Operational Intelligence

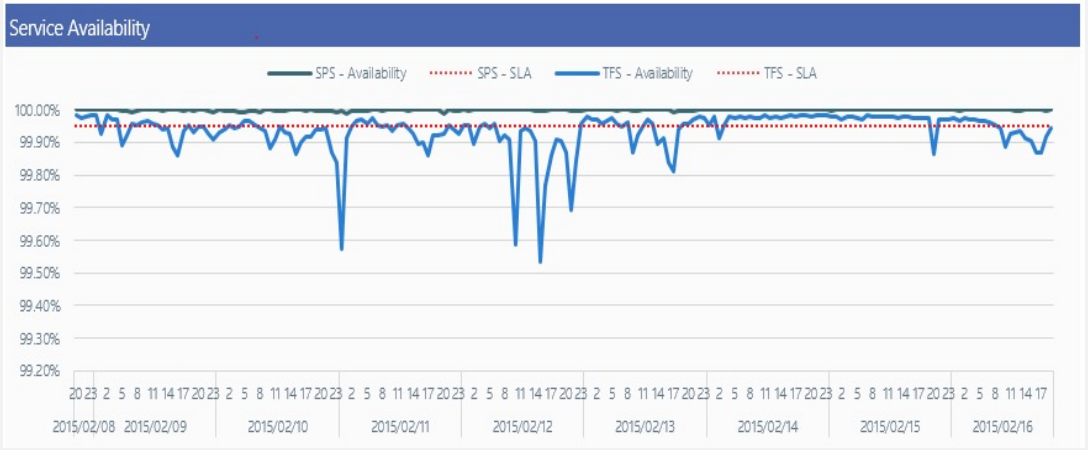
Business Intelligence

Customer Intelligence

KPI metrics

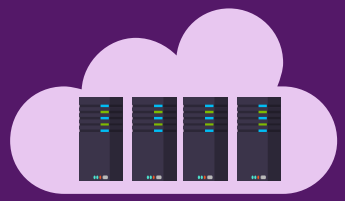
Aggregated metrics that are "cooked" in real-time or in the back-end

Used to calculate availability and generate alerts



Visual Studio Online

Activity Logging | Traces | KPI Metrics | Job History | Customer Intelligence | Synthetic | Perf Counters | Dependent Services



Operational Intelligence

Business Intelligence

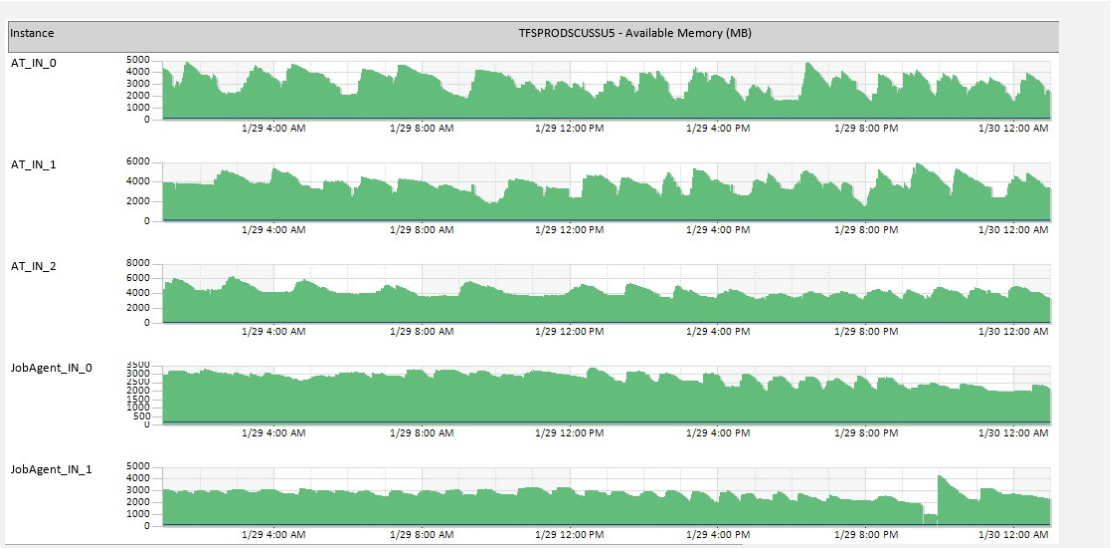
Customer Intelligence

Perf counters

Capture performance of the platform and Azure infrastructure

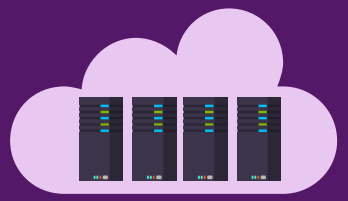
Use for debugging and measuring performance

~50M events per day



Visual Studio Online

Activity Logging | Traces | KPI Metrics | Job History | Customer Intelligence | Synthetic | Perf Counters | Dependent Services



Operational Intelligence

Business Intelligence

Customer Intelligence

Ping mesh

Capture a baseline of network performance and health

Helps to narrow down network related problems quickly – problems external to the system are harder to identify and take longer to escalate

	es2prod-L01	es2prod-L02	T02-C01-A01	T02-C01-A02	T02-C01-A03	T02-C01-A04	T02-C01-A05	T02-C01-A06	T02-C01-A07	T02-C01-A08	T02-C01-A09	T02-C01-A10	T02-C01-A11	T02-C01-A12	T02-C01-A13	T02-C01-A14	T02-C01-A15	T02-C01-A16	T02-C01-A17	T02-C01-A18	T02-C01-A19	T02-C01-A20
es2prod-l01		1	2	2	2	2	37	2	148	2	2	2	2	2	2	2	2	2	3	2	2	11
es2prod-l02	1		4	2	2	2	36	2	119	2	2	2	2	3	2	2	2	2	2	2	3	7
t02-c01-a01	2	3		3	5	4	35	4	122	3	3	3	4	4	3	4	4	3	4	4	4	8
t02-c01-a02	2	2	3		4	3	34	3	347	3	3	3	3	3	4	3	3	3	4	3	4	9
t02-c01-a03	2	2	4	4		3	33	4	119	3	4	4	4	4	3	5	3	4	5	3	5	14
t02-c01-a04	2	3	3	4	4		35	4	122	3	3	4	4	5	3	4	3	4	3	3	3	9
t02-c01-a05	18	26	23	22	21	27		20	119	26	24	20	27	24	25	25	31	19	33	19	29	29
t02-c01-a06	2	3	4	4	4	4	35		121	5	4	4	3	4	3	4	5	5	4	4	4	10
t02-c01-a07	1	2	4	3	3	3	96	4		3	4	2	4	2	4	3	5	3	4	2	4	13
t02-c01-a09	2	2	5	3	6	3	36	4	162	4		3	3	4	4	4	3	4	3	3	3	12
t02-c01-a10	1	2	3	2	5	3	39	4	85	3	4		3	3	3	3	3	3	3	3	4	9
t02-c01-a11	2	2	8	3	5	3	40	4	117	3	4	3		3	3	4	4	3	3	3	3	9
t02-c01-a12	2	3	3	3	7	3	37	3	154	3	4	4	4		4	3	5	4	4	3	5	8
t02-c01-a13	2	2	5	3	5	3	37	3	116	3	3	3	3	3		3	3	4	4	4	3	11
t02-c01-a14	2	3	6	3	5	3	37	3	119	4	3	3	3	3	4		3	4	4	2	4	12
t02-c01-a15	2	3	4	4	6	4	35	3	160	4	4	3	4	5	4	3		4	4	3	4	12
t02-c01-a16	1	2	7	3	6	3	34	3	91	3	5	3	3	3	3	3	3		3	2	3	7
t02-c01-a17	2	3	3	3	4	3	37	3	87	3	4	3	3	3	4	3	4	3		3	4	8
t02-c01-a18	2	2	3	3	4	3	39	3	116	3	4	3	3	3	3	3	3	3	3		3	11
t02-c01-a19	2	3	3	4	4	3	39	4	162	3	4	3	3	3	3	3	4	4	3	2		12
t02-c01-a20	11	5	8	10	8	10	44	6	95	6	7	6	6	7	9	12	6	7	6	6	8	
t02-c01-b01	1	1	2	2	3	2	35	2	82	2	3	2	2	3	3	3	2	2	2	2	3	9
t02-c01-b02	2	3	3	4	4	4	33	4	121	3	4	3	4	4	3	3	4	4	3	3	3	7
t02-c01-b03	1	2	5	3	3	3	34	3	364	3	3	3	2	3	3	2	3	3	3	2	4	8
t02-c01-b04	1	2	3	3	3	4	45	3	154	3	3	3	3	4	3	4	3	4	4	3	3	9
t02-c01-b05	2	3	3	3	4	3	34	3	157	4	4	4	4	4	2	3	3	4	4	3	4	9
t02-c01-b06	2	2	6	2	3	2	32	3	400	2	3	3	2	3	2	3	4	3	3	3	3	7
t02-c01-b07	13	13	15	18	12	19	54	16	127	18	14	10	15	21	15	16	16	19	14	18	15	28
t02-c01-b08	11	11	8	7	13	6	43	9	102	7	8	12	9	7	9	7	7	8	10	8	9	12
t02-c01-b09	2	2	3	2	6	3	36	3	172	3	3	3	3	4	3	3	3	3	3	2	3	7
t02-c01-b10	5	3	9	6	5	6	42	9	89	7	7	5	8	6	6	7	6	7	8	5	4	11

Visual Studio Online

Activity Logging | Traces | KPI Metrics | Job History | Customer Intelligence | Synthetic | Perf Counters | Dependent Services

Operational Intelligence

Business Intelligence

Customer Intelligence

Our learnings as a SaaS provider

Live-site culture principles

Site status is always priority 1

Global response team

On-call DRI (Designated Responsible Individual) by area

Service Delivery team in 5 time zones for 24x7

Weekly live site review

Monthly service review

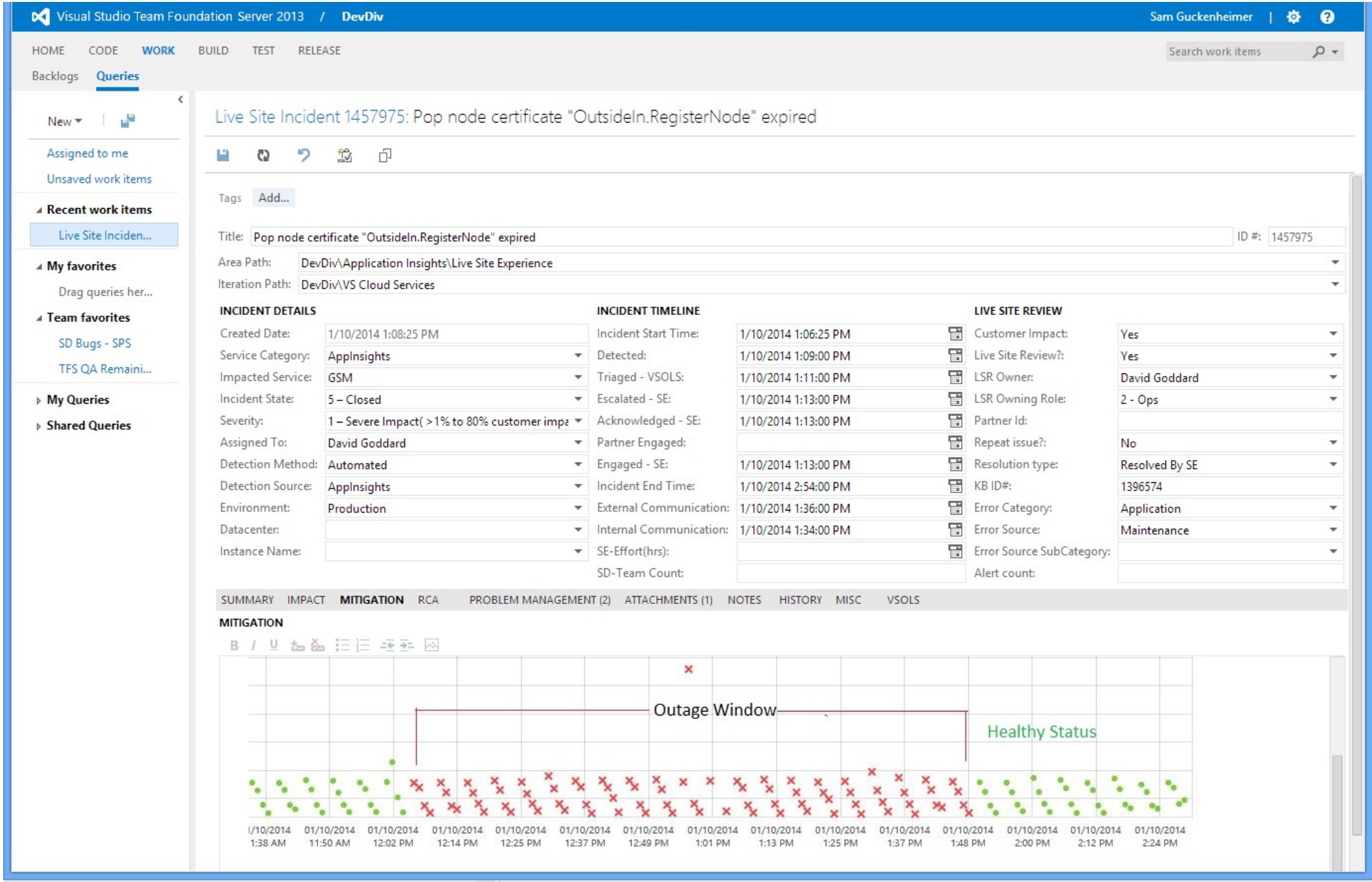
Communicate in real time

Live-site issues become product backlog items

Fix at *root cause*

Automate solutions

Live Site Issues (LSIs)



Auto-dumper

Root Cause

Get diagnostics the first time an issue occurs

Secure

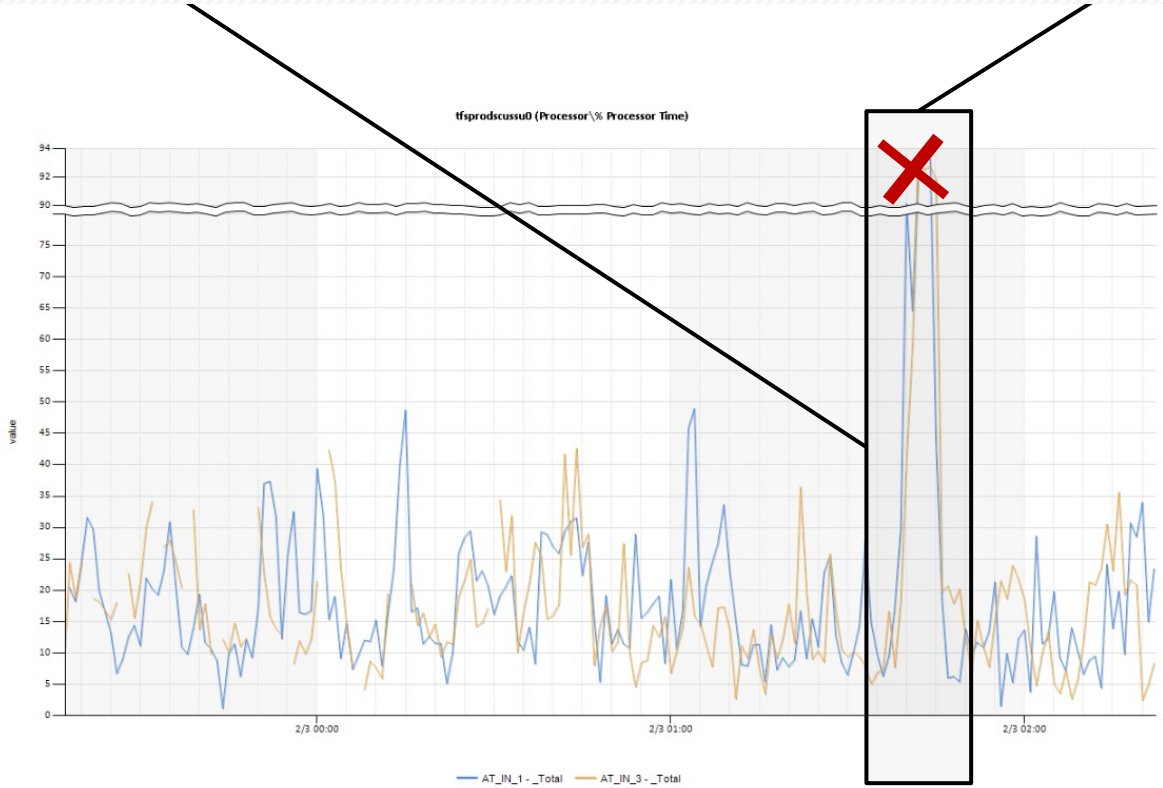
Avoid using RDP which reduces security risk

Fast

Avoid manual repro and data collection

Transient CPU spike. Hard to repro and get diagnostic tools enabled on time (dump, perfview)

Automated trigger can start the diagnostic process automatically and collect data on-demand



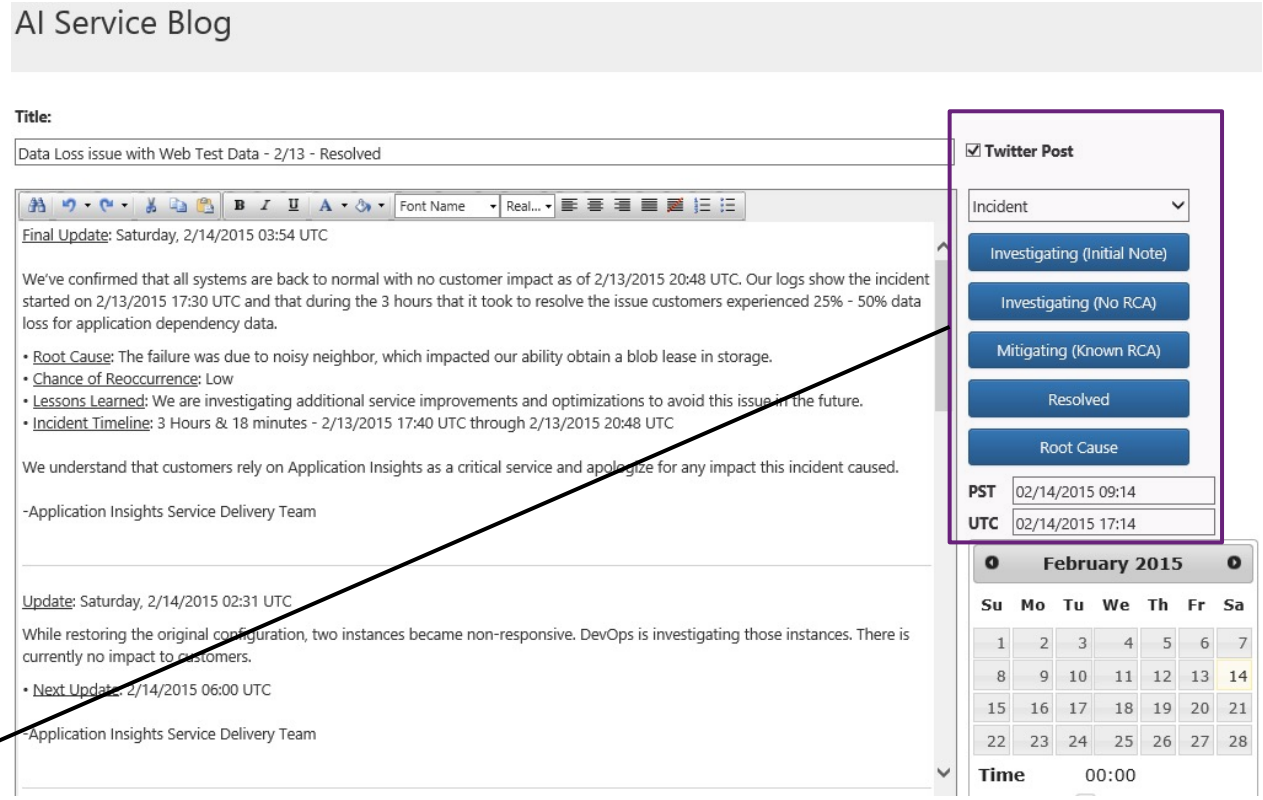
Automating communication

Save time

Automatically post to blog, email and twitter

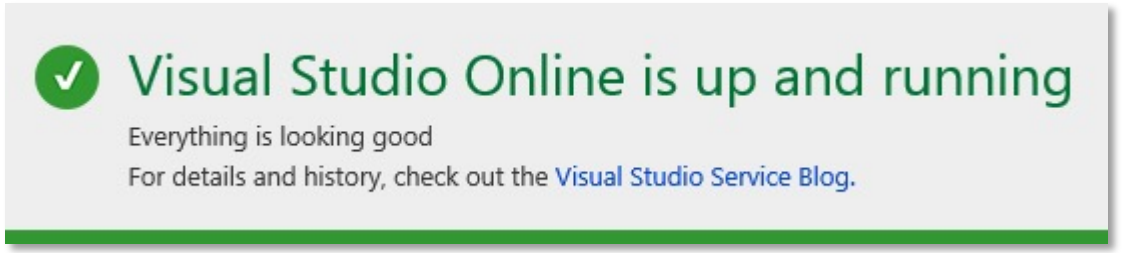
Consistency

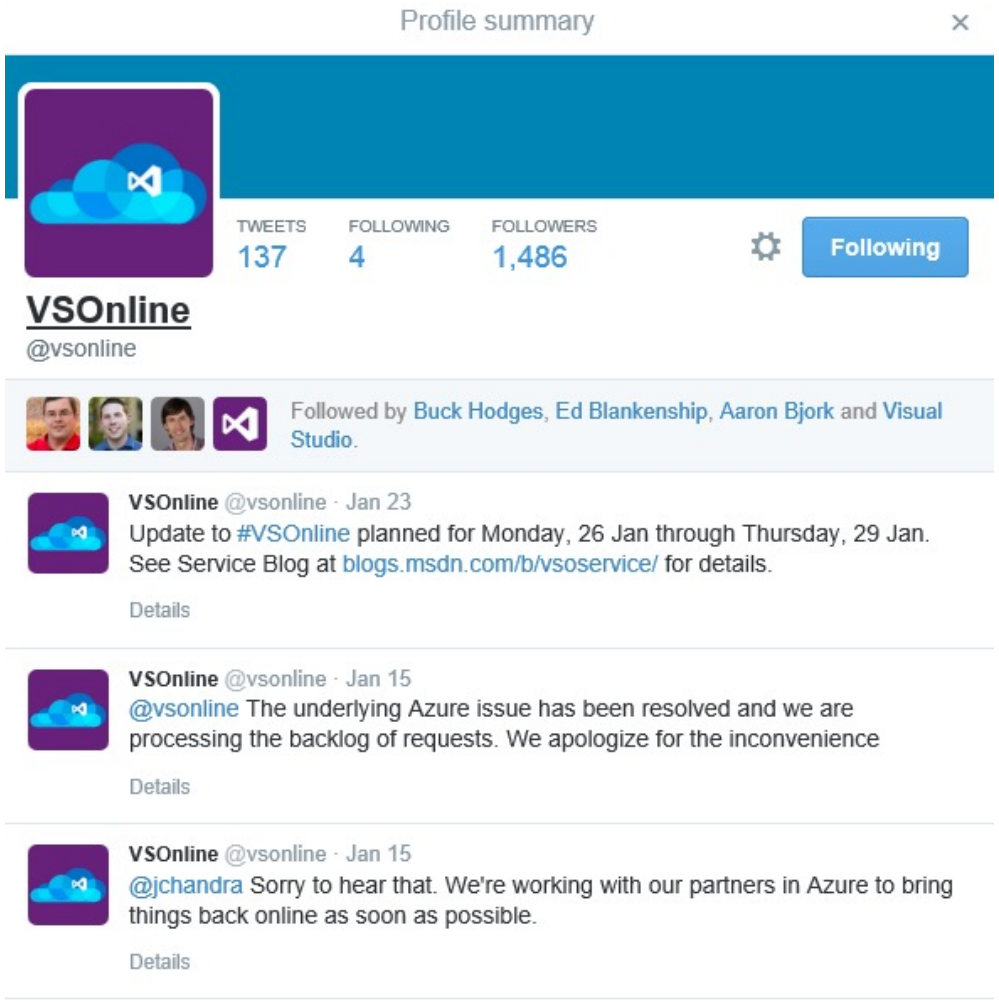
Leverage pre-approved templates

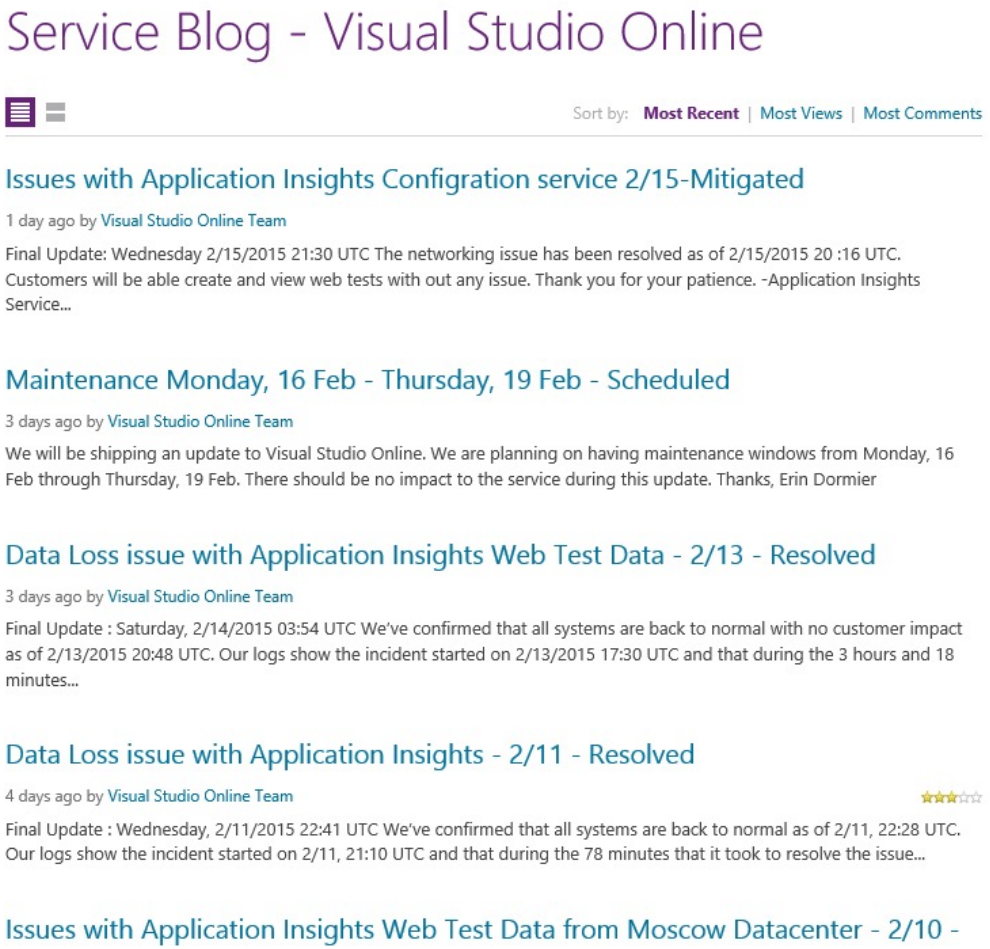


Approved templates make it easy to create an incident notification quickly. During the crisis it also helps the team remember what is required.

Service status visible







RCA (Root Cause Analysis) transparency

A Rough Patch

Brian Harry MS 25 Nov 2013 3:06 PM 10

Either I'm going to get increasingly good at apologizing to fewer and fewer people or we're going to get better at this. I vote for the later.

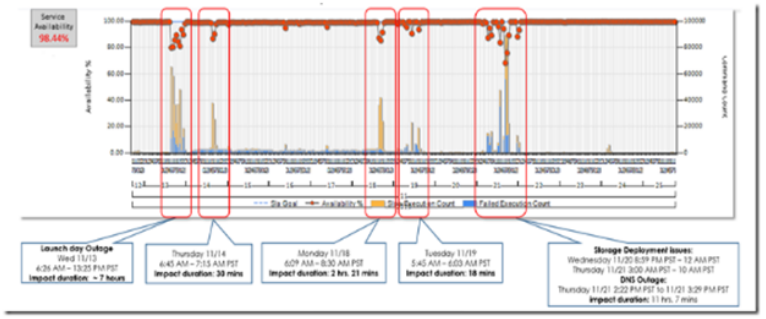
We've had some issues with the service over the past week and a half. I feel terrible about it and I can't apologize enough. It's the biggest incident we've had since the instability created by our service refactoring in the March/April timeframe. I know it's not much consolation but I can assure you that we have taken the issue very seriously and there are a fair number of people on my team who haven't gotten much sleep recently.

The incident started the morning of the Visual Studio 2013 launch when we introduced some significant performance issues with the changes we made. You may not have noticed it by my presentation but for the couple of hours before I was frantically working with the team to restore the service.

At launch, we introduced the commercial terms for the service and enabled people to start paying for usage over the free level. To follow that with a couple of rough weeks is leaving a bad taste in my mouth (and yours too, I'm sure). Although the service is still officially in preview, I think it's reasonable to expect us to do better. So, rather than start off on such a sour note, we are going to extend the "early adopter" program for 1 month giving all existing early adopters an extra month at no charge. We will also add all new paying customers to the early adopter program for the month of December – giving them a full month of use at no charge. Meanwhile we'll be working hard to ensure things run more smoothly.

Hopefully that, at least, demonstrates that we're committed to offering a very reliable service. For the rest of this post, I'm going to walk through all the things that happened and what we learned from them. It's a long read and it's up to you how much of it you want to know.

Here's a picture of our availability graph to save 1,000 words:



Explanation of July 18th outage

Brian Harry MS 31 Jul 2014 5:58 AM 6 RATE THIS

Sorry it took me a week and a half to get to this.

We had the most significant VS Online outage we've had in a while on Friday July 18th. The entire service was unavailable for about 90 minutes. Fortunately it happened during non-peak hours so the number of affected customers was fewer than it might have been but I know that's small consolation to those who *were* affected.

My main goal from any outage that we have is to learn from it. With that learning, I want to make our service better and also share it so, maybe, other people can avoid similar errors.

What happened?

The root cause was that a single database in SQL Azure became very slow. I actually don't know why, so I guess it's not really the root cause but, for my purposes, it's close enough. I trust the SQL Azure team chased that part of the root cause – certainly did loop them in on the incident. Databases will, from time to time, get slow and SQL Azure has been pretty good about that over the past year or so.

The scenario was that Visual Studio (the IDE) was calling our "Shared Platform Services" (a common service instance managing things like identity, user profiles, licensing, etc.) to establish a connection to get notified about updates to roaming settings. The Shared Platform Services were calling Azure Service Bus and it was calling the ailing SQL Azure database.

The slow Azure database caused calls to the Shard Platform Services (SPS) to pile up until all threads in the SPS thread pool were consumed, at which point, all calls to TFS eventually got blocked due to dependencies on SPS. The ultimate result was VS Online being down until we manually disabled our connection to Azure Service Bus an the log jam cleared itself up.

There was a lot to learn from this. Some of it I already knew, some I hadn't thought about but, regardless of which category it was in, it was a damn interesting/enlightening failure.

UPDATE Within the first 10 minutes I've been pinged by a couple of people on my team pointing out that people may interpret this as saying the root cause was Azure DB. Actually, the point of my post is that it doesn't matter what the root cause was. Transient failures will happen in a complex service. The interesting thing is that you react to them appropriately. So regardless of what the trigger was, the "root cause" of the outage was that we did not handle a transient failure in a secondary service properly and allowed it to cascade into a total service outage. I'm also told that I may be wrong about what happened in SB/Azure DB. I try to stay away from saying too much about what happens in other services because it's a dangerous thing to do from afar. I'm not going to take the time to go double check and correct any error because, again, it's not relevant to the discussion. The post isn't about the trigger. The post is about how we reacted to the trigger and what we are going to do to handle such situations better in the future.

Don't let a 'nice to have' feature take down your mission critical ones

I'd say the first and foremost lesson is "Don't let a 'nice to have' feature take down your mission critical ones." There's a notion in services that all services should be loosely coupled and failure tolerant. One service going down should not cause a cascading failure, causing other services to fail but rather only the portion of functionality that absolutely depends on the failing component is unavailable. Services like Google and Bing are great at this. They are composed of dozens or hundreds of services and any single service might be down and you never even notice because most of the experience looks like it always does.

Retrospective on the Aug 14th VS Online outage

Brian Harry MS 22 Aug 2014 11:10 AM 20 RATE THIS

We had a pretty serious outage last Thursday all told it was a little over 5 hours. The symptoms were that performance was so bad that the service was basically unavailable for most people (though there was some intermittent access as various mitigation steps were taken). It started around 14:00 UTC and ended a little before 19:30 UTC. This duration and severity makes this one of the worst incidents we've ever had on VS Online.

We feel terrible about it and continue to be committed to doing everything we can to prevent outages. I'm sorry for the problems it caused. The team worked tirelessly from Thursday through Sunday both to address the immediate health issues and to fix underlying bugs that might cause recurrences.

As you might imagine, for the past week, we've been hard at work trying to understand what happened and what changes we have to make to prevent such things in the future. It is often very difficult to find proof of the exact trigger for outages but you can learn a ton by studying them closely.

On an outage like this, there's a set of questions I always ask, and they include:

What happened?

What happened was that one of the core SPS (Shared Platform Services) databases became overwhelmed with database updates and started queuing up so badly that it effectively blocked callers. Since SPS is part of the authentication and licensing process, we can't just completely ignore it – though I would suggest that if it became very sluggish, it wouldn't be the end of the world if we bypassed some licensing checks to keep the service responsive.

What was the trigger? What made it happen today vs yesterday or any other day?

Though we've worked hard on this question, we don't have any definitive answer (we're still pursuing it though). We know that before the incident, some configuration changes were made that caused a significant increase in traffic between our "TFS" service and our "SPS" (Shared Platform Service). That traffic involved additional license validation checks that had been improperly disabled. We also know that, at about the same time, we saw a spike in latencies and failed deliveries of Service Bus messages. We believe that one or both of these were key triggers but we are missing some logging on SPS database access to be able to be 100% certain. Hopefully, in the next few days, we'll know more conclusively.

What was the "root cause"?

This is different than the trigger in the sense that the trigger is often a condition that caused some cascading effect. The root cause is more about understanding why the effect cascaded and why it took the system down. It turns out that, I believe, the root cause was that we had accumulated a series of bugs that were causing extra SPS database work to be done and that the system was inherently unstable – from a performance perspective. It just took some poke at the system – in the form of extra identity or licensing churn to cause a ripple effect on these bugs. Most, but not all, of them were introduced in the last few

Precise alerting is key to fast detection …

… but poor alerting makes DevOps unhappy

Redundant alerts for same the issue

Needed to set right thresholds and tune often

Stateless alerts contributed to further noise

We set alerting goals

Every alert must be actionable and represent a real issue with the system

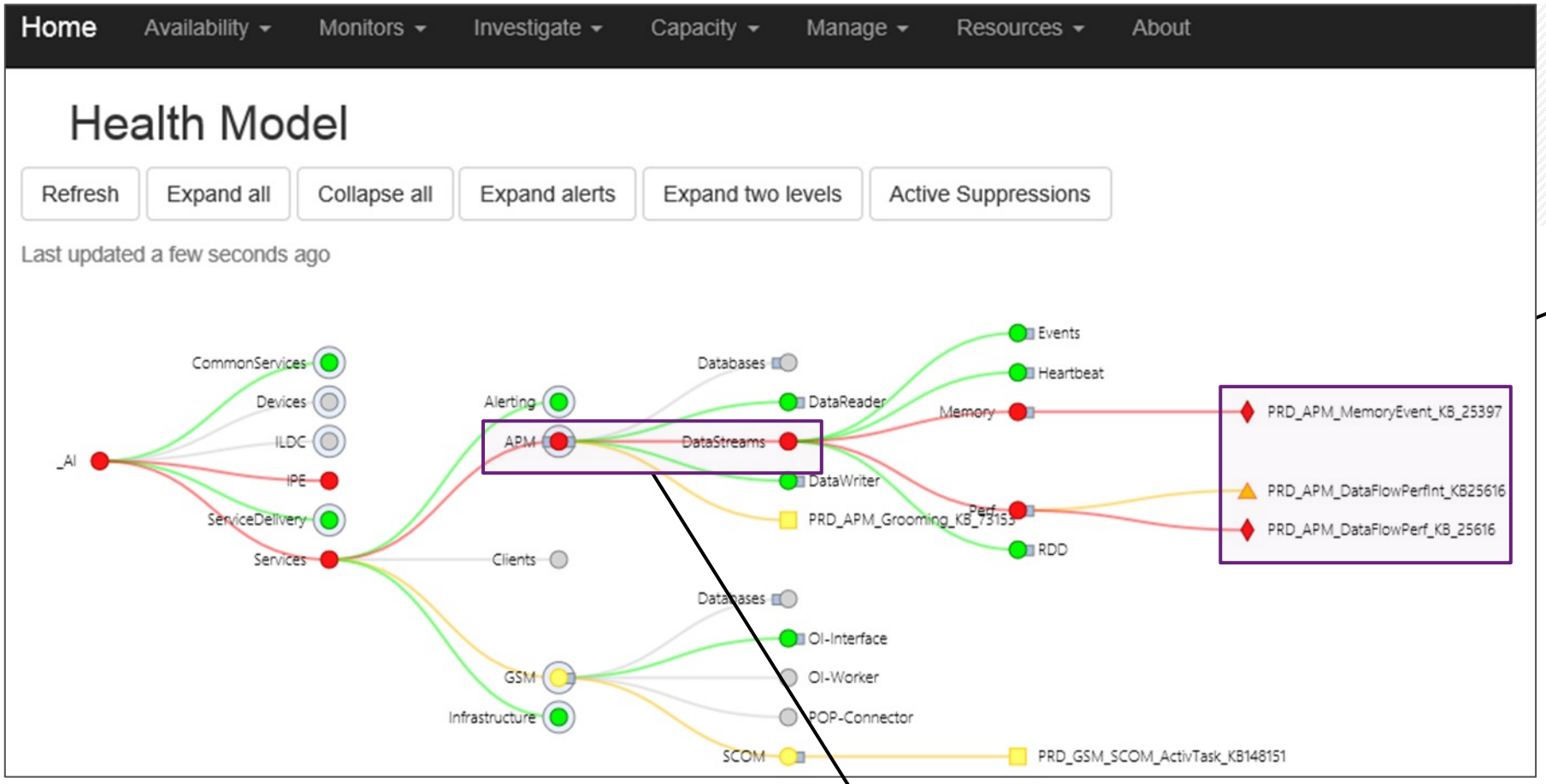
Alerts should create a sense of urgency – false alerts dilute that

Solution

Consolidate alerts so that only actionable alerts are sent to team

Autoroute according a health model based on suspect route cause

Health model in action



Found 3 errors for memory and performance

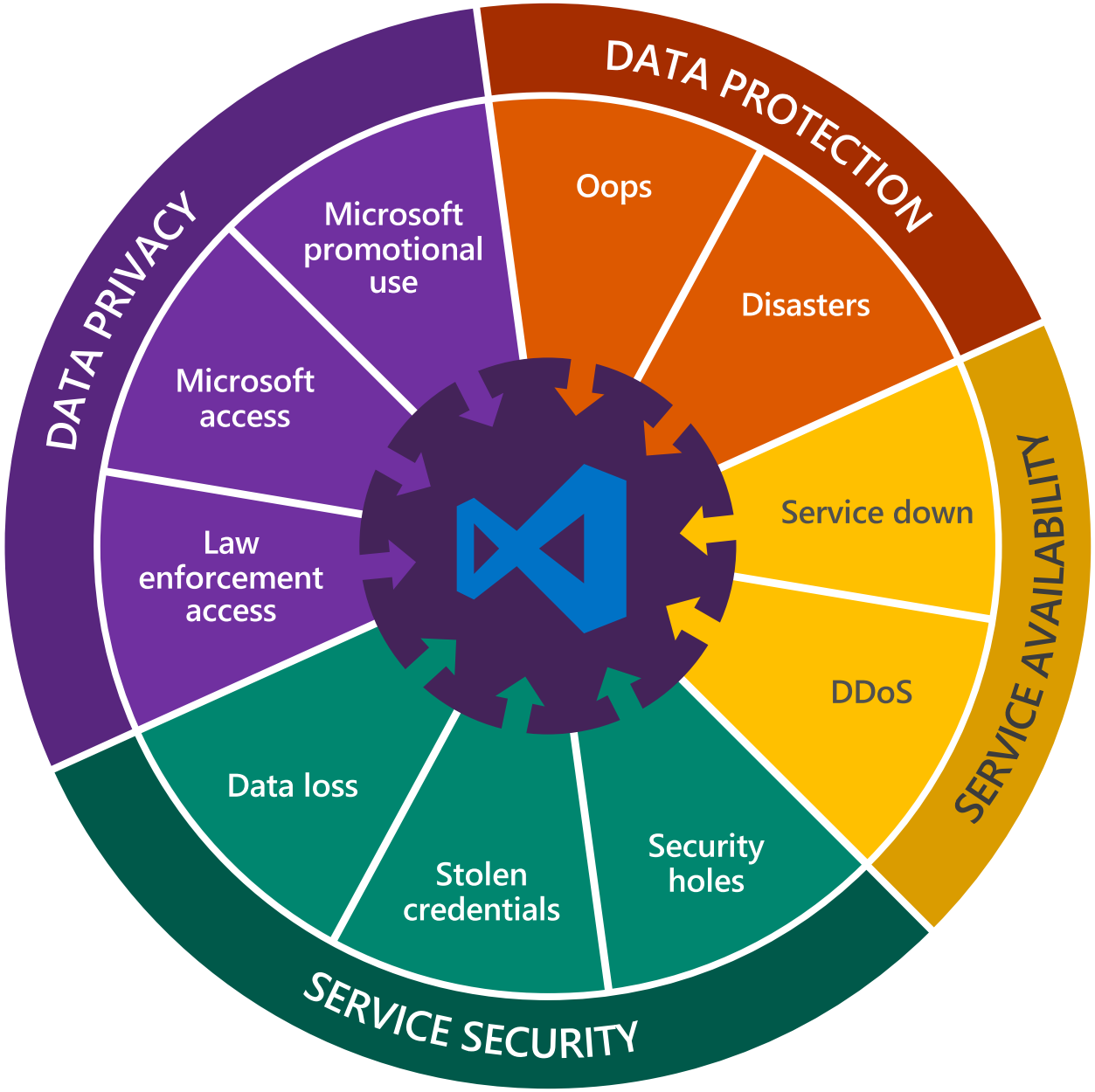
All 3 errors are related to the same code defect

APM component mapped to feature team
Auto-dialer engaged Global DRI

Eliminated alert noise:
~928 alerts per week to ~22
Reduced DRI escalations by ~56%

Our security mindset is **assume breached**, not just **prevent breach!**

Comprehensive security program designed with this mindset



Secrets management

Secrets

Certificates
Storage keys
Passwords
Service accounts

Challenges

Expire
Expiry date associates with all secrets

Leak
Accidental exposure of secrets

Theft
Hacking attempts and security breaches

How we do it

Azure secret store
Encrypted store w/ self-service API

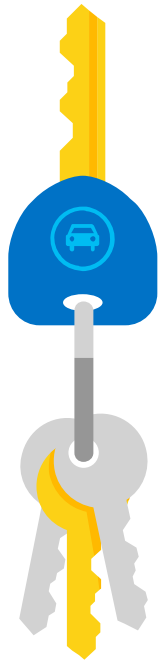
Encrypted config
Encrypt secrets in app config

Expiring object scanner
Scan and alert on secret expiry date

Rotate
Change secrets regularly

Audit
Alert on unauthorized access

1000+
Secrets in VSO



Bad day

Brian Harry MS | 23 Feb 2013 4:29 AM | 11 | RATE THIS

Clearly yesterday was a bad day. Team Foundation Service was mostly down for approximately 9 hours. The underlying issue was an expired SSL certificate in Windows Azure storage. We use HTTPS to access Windows Azure storage, where we store source code files, Git repos, work item attachments and more. The expired certificate prevented access to any of this information, making much of the TFService functionality unavailable.

We were watching the issue very closely, were on the support bridge continuously and were investigating options to mitigate the outage. Unfortunately we were not successful and had to wait until the underlying Azure issue was resolved. I have a new appreciation for the "fog of war" that happens so easily during a large scale crisis. We'll be sitting down early this week to go through the timeline hour by hour – what we knew, what we didn't know, what we tried, what else we could have tried, how we communicated with customers and everything else to learn everything we can from the experience.

Our learnings as a SaaS provider

Automating deployments

Deploy anytime

Online operation – service stays up 24x7x365

Fully automated through TFS Release Management

Multiple versions must coexist peacefully

Staging

Canary stamp first (SU0), then other instances

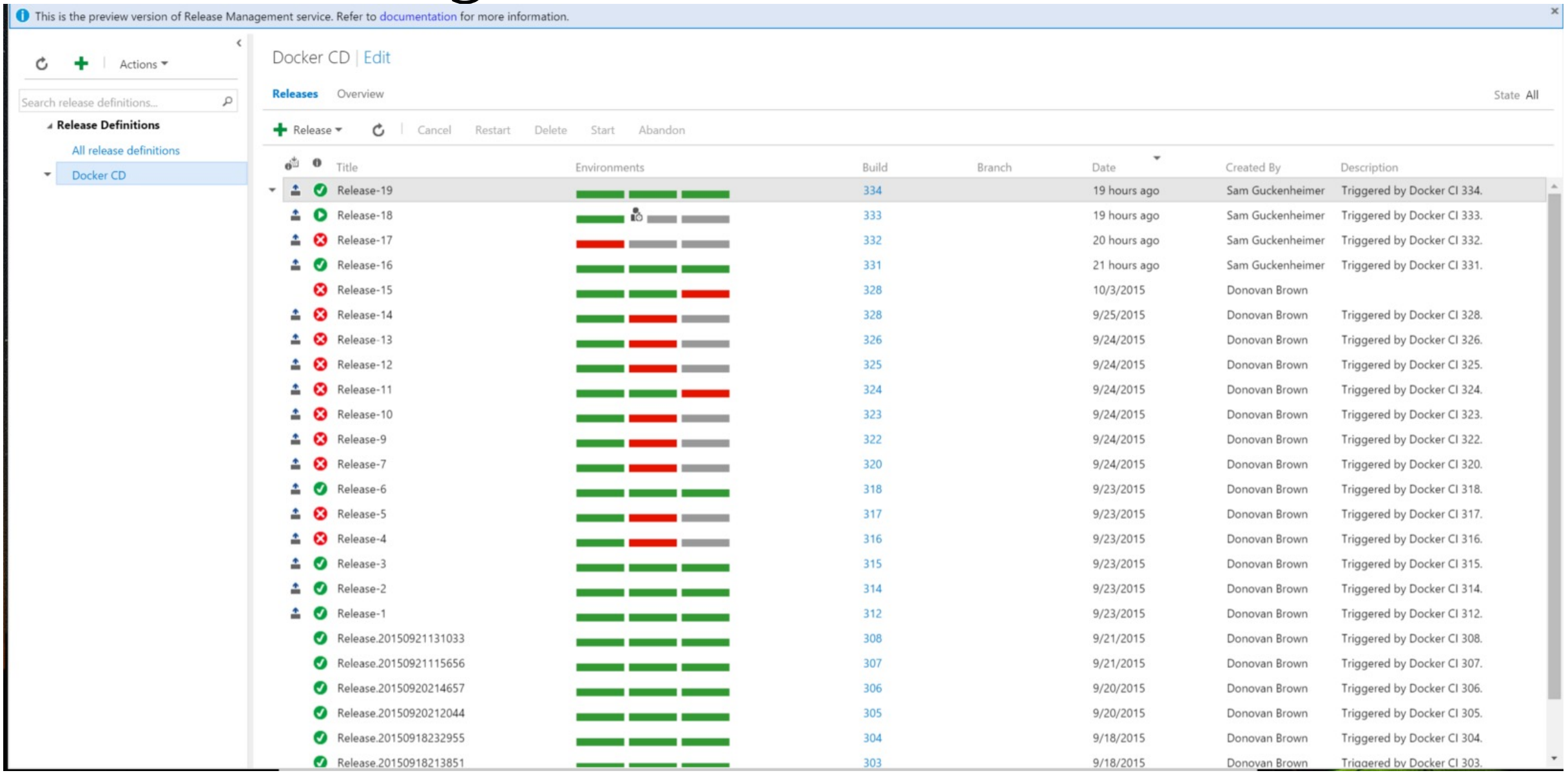
Detailed health analysis after each stamp

Exposure control using feature flags

Enable or disable features at runtime without code changes or redeployment

Progressive disclosure of features

Release Management



Feature flags

All code is deployed, but feature flags control exposure

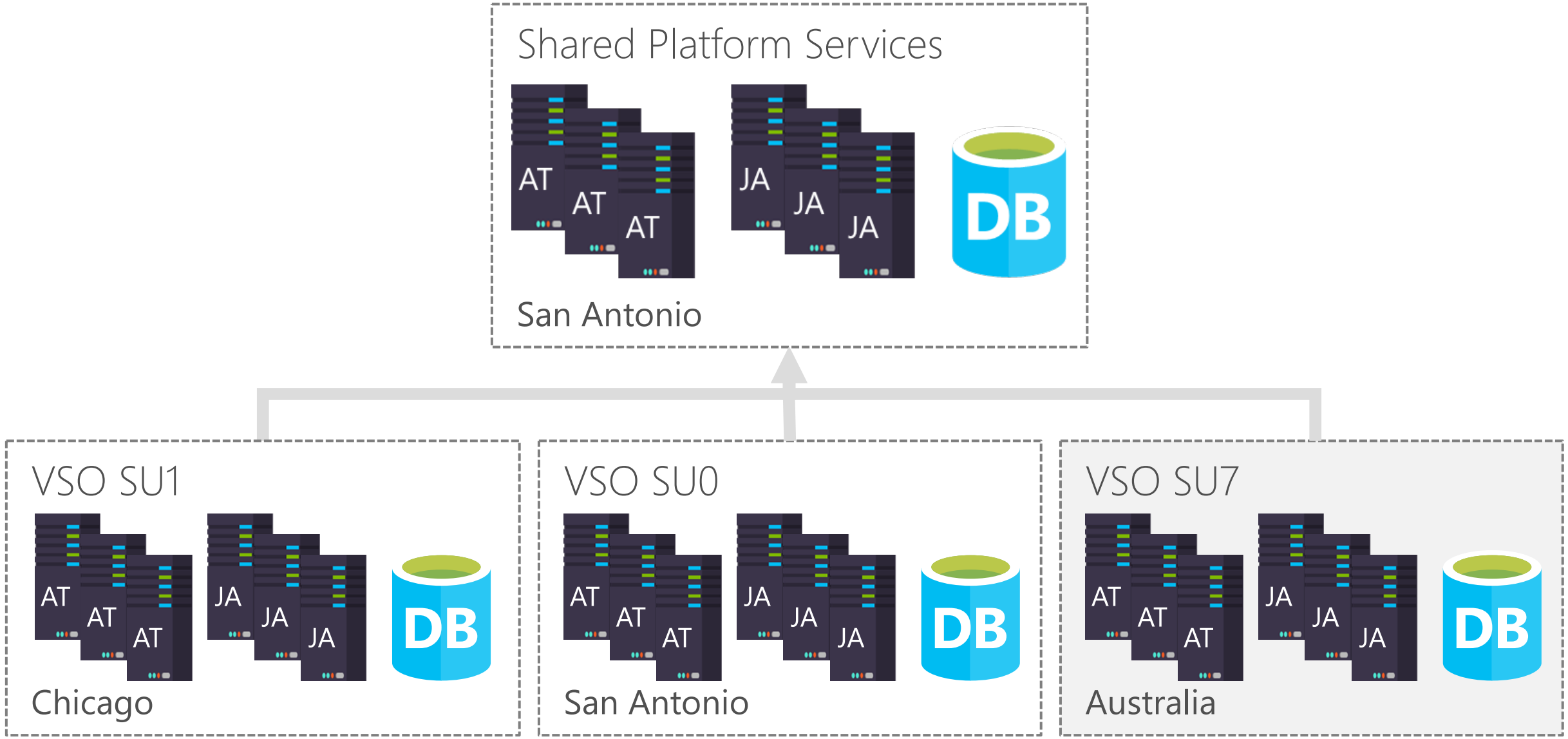
Flags provide runtime control down to individual user

Users can be added or removed with no redeployment

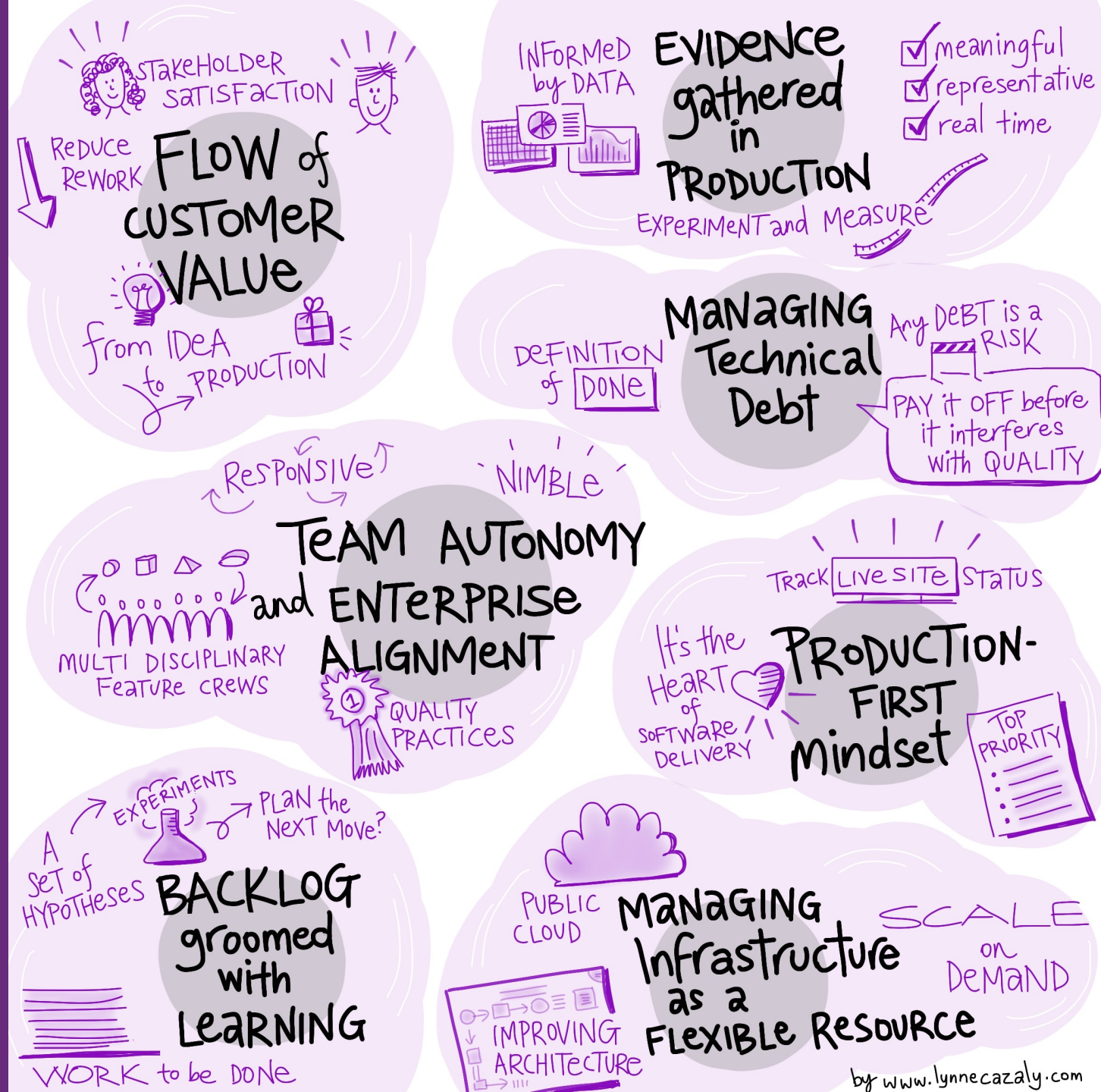
Enables dark launch

Mechanism for progressive experimentation & refinement

Multiple scale units enable canarying



Our learnings as a SaaS provider



Metrics we track

Feature Team	Livesite Health and Debt											
		% of LSIs automatically detected		Max TTM (mins)		# LS prevention work items		LSPs rank <= 50 & P2 or higher LSI work > 21d		# of DTS over SLA	
Modern WIT		NA		NA		24		15		6	
WIT IQ		NA		NA		7		3		0	

Feature Team	Engineering Debt									
		# Bugs resolved/ engr in last 21 days		# Active bugs per engr		# P0 or P1 bugs > 21 days	Team Azure NAR % pass	Team Azure NAR time (mins)	Test Improvement		Security: # WI > 21 days	
Modern WIT		11		8		8	92.5%	89	2		2	
WIT IQ		6		3		5	94.6%	159	4		3	

Live Site Health

Time to Detect

Time To Mitigate

Incident prevention items

Aging live site problems

Customer support metrics

(SLA, MPI, top drivers)

Velocity

Time to build

Time to self test

Time to deploy

Time to learn

(Telemetry pipe)

Engineering

Bug cap per engineer

Aging bugs in important categories

Pass rate & coverage

Usage

Acquisition

Engagement

Retention

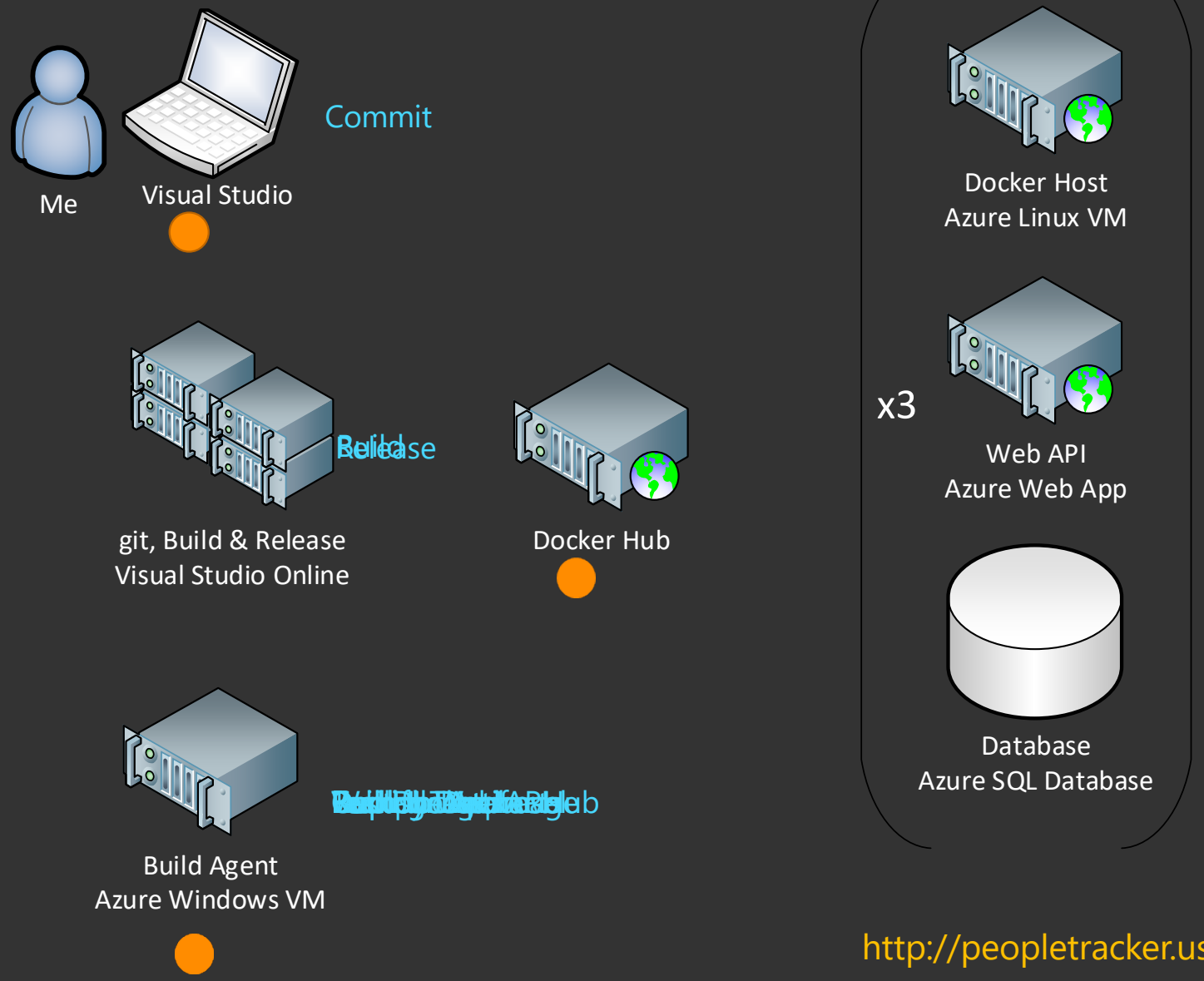
Conversion

Churn

DEMO: Productizing Our Learnings

Automate Deployment with Release Management

Containers across Windows & Linux



http://peopletracker.us

There’s no place like production!