I’m an SRE Lead !

Now What ?

Ritchie Schacher
STSM, SRE Architect, Bluemix DevOps Services

Rob Orr
Program Director SRE, DevOps & Analytics Services

Introductions

Ritchie Schacher
STSM SRE, Architect
Bluemix DevOps Services

Ritchie Schacher is a Senior Technical Staff Member with IBM® Bluemix® DevOps Services, and lead architect for SRE. He has a background in developer tools and team collaboration. He has broad experience in all aspects of the software development lifecycle, including product conception, planning and managing, design, coding, testing, deploying, hosting, releasing, and supporting products.

For the last 3 years he has been a part of an exciting team developing and supporting cloud-based SaaS offerings for Bluemix®. In his spare time, Ritchie enjoys playing classical guitar.

Rob Orr
Program Director
SRE, DevOps & Analytics

Rob Orr is a Program Director at IBM Cloud and leads the SRE team responsible for BlueMix® DevOps Services. These services include developer tools for Toolchains, Pipline, 3rd party integrations and Git repository services. His current projects include Automation tooling, Developing SLI's, and predictive monitoring.

Mr. Orr joined IBM in 1998 holding various positions throughout his career with IBM including leadership roles in Audit, Security, Service Management and Operations. Rob holds a bachelor's degree in Computer Science from the University of Maryland, and holds 5 patents and has published papers in the IBM systems Journal on IT Standards and Systems Management

Agenda

Introductions

Problem Statement

Why we chose the SRE model

- Motivations
- Working model

Staged implementation approach

Avoiding pitfalls and lessons learned

Questions

Problem Statement

So, your organization has decided to build an **SRE team**, and you are one of the leaders. You have multiple stakeholders with different expectations and agendas. Your **executive team** expects early and immediate results. They want checklists, goals, and roadmaps, and they want it **now**. Peer dev managers may believe that you now have a magic wand that will make their headache's disappear, or worse, may believe that they can offload all the dev team's **dirty work** to you.

Are you overwhelmed, and wondering how to get started? How will you organize? How will you **manage expectations?** What management style should you use? How will you define and measure success?

In this session, we will cover a phased approach to organize your SRE team based on our real-world experience operating our cloud-based SaaS DevOps toolset. We will also cover some of our lessons learned and pitfalls for you to avoid to help you on your SRE journey.

SRE Motivations

Why SRE? We had a working DevOps model

Improve **availability** of our services

Ensure consistency and discipline in **processes** for incident management and response

Improve incident **MTTA, MTTR**

Ensure separation of concerns from development priorities

- Leadership **Discipline**
- Ensure SRE work is budgeted, prioritized
- Manage productivity and avoid **burnout**
- Enhance resource retention

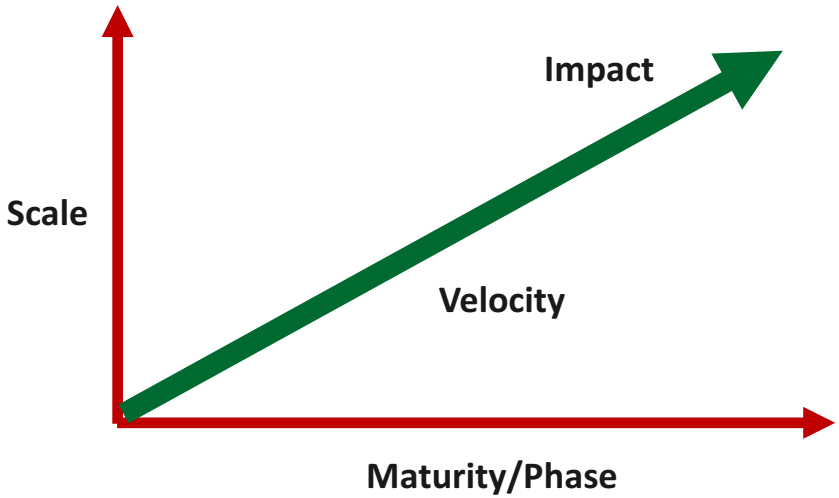
Ensure separation of duties and compliance

Provide economy of scale by having **specialized** engineers focused on:

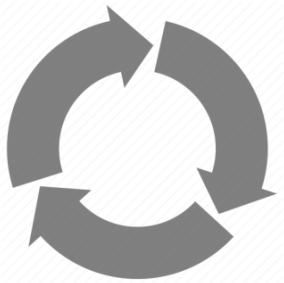
- availability, monitoring, systems management, and
- Security, compliance, and BC/DR

A staged approach to implementation

1. Define Scope and Obtain Stakeholder Buy-in
2. Define Tools, Processes, and Backlog
3. Build and Organize the SRE Team
4. Implement and Evolve



Iterate along the way



Define Scope and Obtain Stakeholder Buy-in

Build the leadership team and the mission

- SRE Technical leads, Service Managers, Architects
- Subject matter experts
- Define core values

Define the roles and responsibilities (SRE - Dev)

Define the Benchmarks

- Scorecards and checklists
- Promotion gates and approvals

Ensure stakeholder buy-in

- Executives
- Peer service managers
- Service tech leads

Example: SRE Values

We are **obsessed** with availability and reliability

We are an **Engineering** team

- We promote a high-performance culture
- We fill our skills gaps as necessary
- We have an automation mindset
- We use metrics, monitoring and measurement to ensure accountability
- We use Agile to manage our workload

We shape direction and **Positively Influence Outcomes** for our services

- Our RCA actions, checklists, and gap analyses directly feed into development prioritization

We **partner** with the service teams in all respects

- We understand and can use our services and capabilities
- We focus on the customer experience
- We discourage an ”us vs. them” mindset
- We align our priorities

We respect the **autonomy** of the service squads

Example: SRE Responsibilities and Control Points

Responsibilities

- Maintain Stated Availability of Service
- Incident, Problem, Capacity, Security, and Change Management
- Development of features that enhance Site Availability
- Automation and Monitoring Development
- Creation and Publishing of Dashboards and Availability Metrics

Control Points

- Provide Architectural Review and Signoff on a Service based on ability to achieve availability targets
- Accept or Reject services based on their ability to achieve SLA
- Own the Pipeline and Deployments to production
- Gate changes into a production Service contingent on error budget
- Use Error Budget to balance development prioritization between SRE content and new features

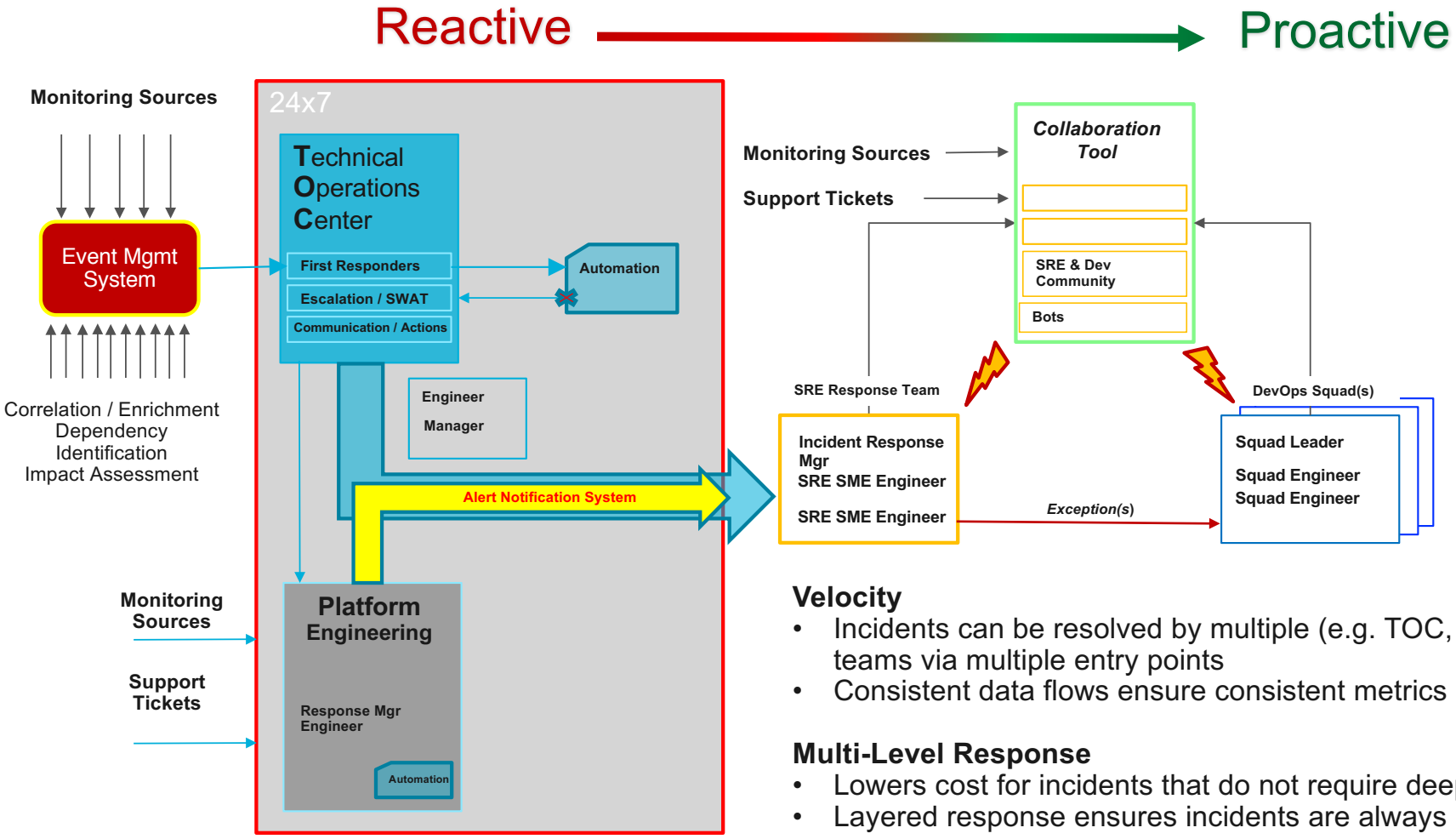
Decisions and recommendations of the SRE team are independent of functional or date commitments

Define Tools, Processes, and Backlog

Know, define or redefine tools and processes

- Tracking and Planning
- Monitoring and Availability Reporting
- Recovery Procedures & Automation
- Incident Management
- RCAs (with traceability and reporting)
- Deployments
- Test reporting
- Collaboration

Example: Incident Operational Model



Velocity

- Incidents can be resolved by multiple (e.g. TOC, SRE, Dev) teams via multiple entry points
- Consistent data flows ensure consistent metrics

Multi-Level Response

- Lowers cost for incidents that do not require deep skills
- Layered response ensures incidents are always handled

Define Tools, Processes, and Backlog

Establish **tracking and planning first**; everything else becomes a prioritized work item

Organize SRE **backlog**

Ensure prioritization of SRE work items is **consistent** with feature/function of service

Determine **control points** for adjusting work priorities, e.g.

- 7 day / 30 day availability **metrics**
- Business priorities of the service
- Security compliance

Establish **collaboration** communication and consistent information sharing

- Where do you go to find specific information types?
- Expedite information access and team **communication** during incident recovery

Build and Organize the team

Build the **engineering** team
- New hires or rotational assignments?
- Evaluate candidate skills and experience
- Ensure broad skill coverage across all aspects of SRE

Organize vertically and horizontally: by **service** and by **discipline**

Designate **SMEs** for each service
- Avoid Human SPoF

Develop domain **expertise** (in the services)
- Architecture and topology, development environments, microservices, deployments
- Acquire skills to develop SRE enhancements to service

Define and **train** call out groups for incident resolution

Cultivate the SRE **mindset**
- Leadership coaching, all hands meetings, reinforce SRE values, conduct retrospectives

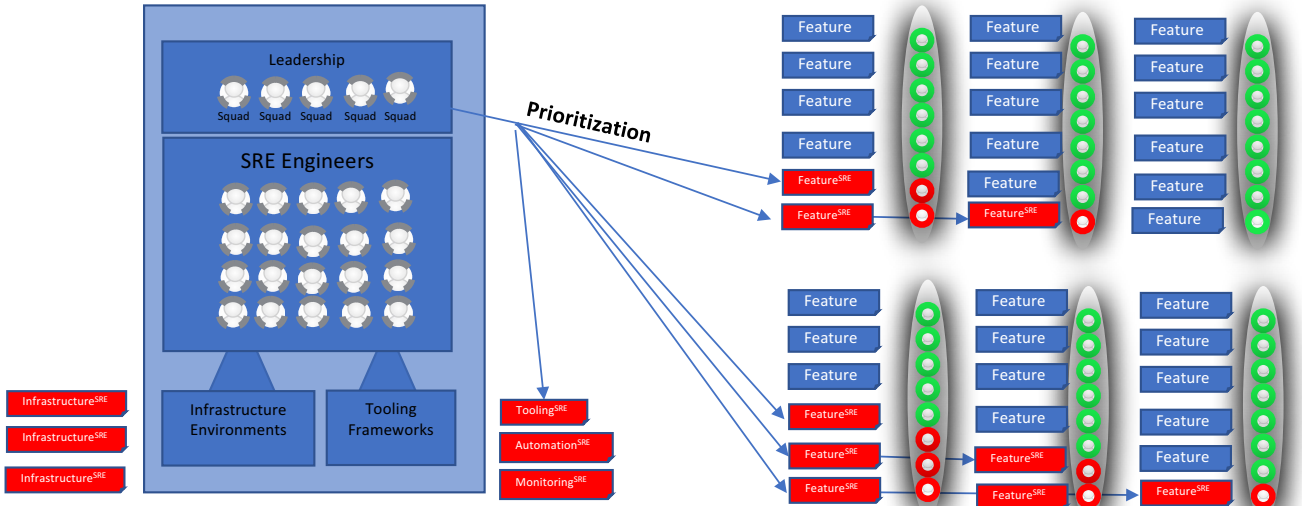
SRE Squad to Service Squad Operational Model

SRE Squad

- SRE STSM drives org-wide priorities
- Engineers rotate on flexible cadence between SRE and Feature squads (some core remain)

Service Development Squads

- Levels of maturity drive amount of resource required for SRE backlog
- 90% of the time these team members are the ones making code changes to the service



SRE Backlog examples

Automation Framework
Self Service Infrastructure
Logging & Metrics
Monitoring & Reporting
Scaling

Feature Squad SRE Backlog examples

High Availability
Rate Limiting
Zero Downtime Deployment
Resiliency
Serviceability

Implement and Evolve

Used a **phased implementation** for SRE ownership of a Service

- **Level I:** Monitoring and reporting tool frameworks, first responder service, Incident Manager support, RCA leadership
- **Level II:** Incident Mgmt, Deployment Mgmt, Availability, Security, Compliance and BC/DR

Onboard services to entry level of SRE ownership (Level 1)

Conduct **gap analysis** of **services** to:

- Assess the maturity of the service
- Identify gaps gating full SRE take over

Define a **roadmap** of **activities** required for **SRE Ownership** and collaborate with service development to build a plan to address

Define service **SLOs and SLIs**

Implement SLI **monitoring and reporting**

SRE Take over of services (Level II **Full SRE Ownership**)

Avoiding Pitfalls and Lessons Learned

Manage expectations

- Agree on the mission and scope first
- Keep it grounded in reality (e.g. development is NOT off the hook)
- Review and negotiate the roles and responsibilities
- Get **buy-in** from **everyone**

Don’t throw an army of engineers at the problem before you have your mission, scope and backlog defined

- Engineers will appreciate a plan and structure with clearly communicated expectations
- Executives and peer development managers will know what their role is and what to expect

Crawl, Walk, Run

- Tackle the obvious items first (e.g RCAs, monitoring gaps, runbooks, processes, reducing MTTA/MTTR)
- Show early success, incremental progress
- Build on success with improved metrics and automation

Ruthlessly prioritize the backlog

- Better to select a handful of items to focus on and do well, than try to do everything with slow velocity
- Review the priorities monthly with stakeholders
- Practice Agile!

Avoiding Pitfalls and Lessons Learned

DevOps and SRE are more than just rebranding of Dev and Operations

- Push the engineers out of an ops mindset, comfort zone
- Watch for an ”us vs. them” culture
- Be pragmatic vs. purist, without sacrificing principles
- Push for automation

SRE doesn’t come for free

- This is a focus that did not exist before; either pull resources from services or hire new
- Short term velocity will be slow but will improve
- Good thing is that once established, this approach scales

Not enough engineers, not right skills

- SRE must be seeded with engineers experienced in the services
- Establish a resource rotation model with development to balance out the team
- Negotiate with the stakeholders; know who they are sending over

Questions

Thank you!