Site Reliability Engineering

TACS

Service Management

- **Service Management**
  - Todo lo que **no** es **desarrollo de features** del sistema
  - **Mantenimiento**

Service Management

- **Sysadmin approach**
  - Costos
    - **directos** por intervención manual, **ops** crece junto al sistema
    - **indirectos** por gap entre development y ops
      - skill set
      - terminología
      - suposiciones sobre estabilidad de producto
  - **Tensión** entre **objetivos** de **development** y **ops:**
    - **generar cambios VS estabilidad**
  - Estos objetivos en **extremo:**
    - *-Dev: "queremos deployar cualquier cosa, en cualquier momento sin obstáculos"*
    - -Ops: "No queremos cambiar nada en el sistema una vez que ya esté andando"

DevOps

- *Conjunto de prácticas destinadas a reducir time to market, garantizando alta calidad*

DevOps

- *Principios*
  - Reducir silos en la organización
  - Aceptar el fallo como normal
    - Reducir ATTR
    - Reducir ATTF
  - Implementar cambios graduales. Mayor frecuencia de deploy
  - Apalancar tooling y automatización
    - Deploys
    - Migraciones / mantenimientos
    - Enlarges / shrinks
  - Medir todo

Service Reliability Engineering

- *SRE es una implementación de DevOps*
  - super opinionated, concrete implementation of DevOps
  - Concrete set of practices
  - Consistent and optimized way of implement devops principles

Service Reliability Engineering

- ***Reliability*** *es el feature más importante*
  - sin confianza, no hay usuarios, sin usuarios, el sistema no tiene valor
  - nuestros usuarios son los que miden la **confiabilidad** del sistema, no su monitoreo/logs
  - Que pasa si el sitio se cae?
    - Pérdida inmediata
      - Bajan las ventas
    - Pérdida intangible
      - Costo de imagen
      - Migración a la competencia

Service Reliability Engineering

- ***Principio clave de SRE***
  - *100% Disponible es la confiabilidad equivocada para básicamente todo*

Availability level	Allowed unavailability window					
	per year	per quarter	per month	per week	per day	per hour
90%	36.5 days	9 days	3 days	16.8 hours	2.4 hours	6 minutes
95%	18.25 days	4.5 days	1.5 days	8.4 hours	1.2 hours	3 minutes
99%	3.65 days	21.6 hours	7.2 hours	1.68 hours	14.4 minutes	36 seconds
99.5%	1.83 days	10.8 hours	3.6 hours	50.4 minutes	7.20 minutes	18 seconds
99.9%	8.76 hours	2.16 hours	43.2 minutes	10.1 minutes	1.44 minutes	3.6 seconds
99.95%	4.38 hours	1.08 hours	21.6 minutes	5.04 minutes	43.2 seconds	1.8 seconds
99.99%	52.6 minutes	12.96 minutes	4.32 minutes	60.5 seconds	8.64 seconds	0.36 seconds
99.999%	5.26 minutes	1.30 minutes	25.9 seconds	6.05 seconds	0.87 seconds	0.04 seconds

Service Reliability Engineering

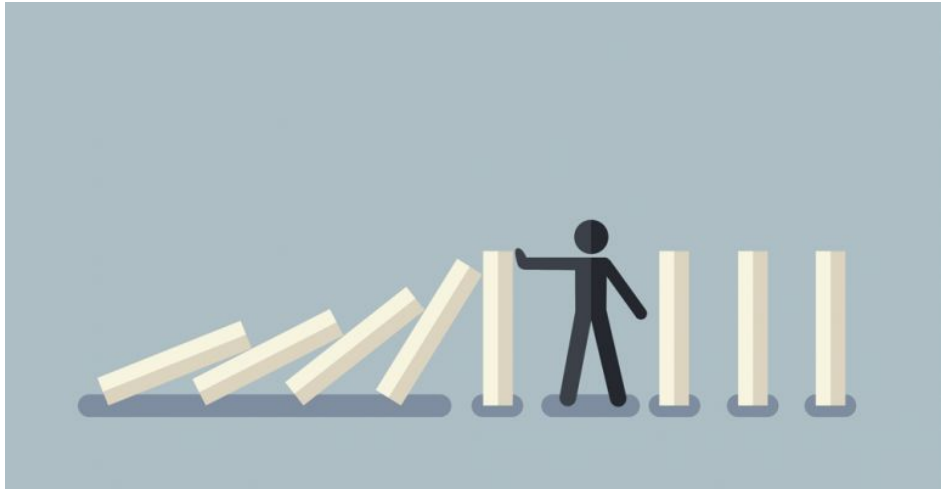
- ***Error Budget***
  - Product Management y SRE establecen Availability Target
  - Error budget (unavailability) = **100% - Availability Target**
  - **Control loop** para utilizar el budget
    - speed up
    - slow down

Service Reliability Engineering

- ***Términos***
  - SLI
    - Service Level Indicator
    - Una medición bien definida de **successful enough**
    - Usado en la definición de SLO y SLA
    - func (metric) < threshold
  - SLO
    - Service Level Objective
    - target de interacciones **exitosas**
    - lower bound ≤ **SLI** ≤ upper bound
  - SLA
    - consecuencias
    - goal
    - SLA = SLO + margen

Service Reliability Engineering

- *Error budgets y SLO previenen la **fatiga de intuición***
- *Si supero el error budget debemos manejar el riesgo:*
  - *freeze*
  - *trabajar en minimizar el riesgo*
    - *ATTR*
    - *ATTF*



Service Reliability Engineering

- *SLO*
  - *Si no tengo un objetivo claro puedo hacer **sobreingeniería** y quizás no lo vale*
  - *Si se a donde quiero llegar se que tan drástica tiene que ser la solución o cuánto esfuerzo estoy dispuesto a invertir*
  - *El **SLO** no es independiente al negocio, sino que se calcula en base al daño que se puede soportar*
  - *Es distinto para cada servicio*

Cómo empezar a implementar SRE?

Cómo empezar con SRE

- ***Voluntad***
- ***Antipattern: anónimo***
  - ***Tratar de implementar sin un sponsor ejecutivo***
  - ***(sponsor ejecutivo != mandato ejecutivo)***

Cómo empezar con SRE

---

1. ***Una aplicación a la vez***
    - ***Qué entiendo por “aplicación”?***
    - ***App = Discrete failure domain***

Cómo empezar con SRE

---

2. ***Empezar definiendo el Error Budget***
    - *Si hacés esto primero, el resto de las cosas irán funcionando por sí mismas*
    - *Se desprende del* ***SLO,*** *nos indica si podemos* ***acelerar o frenar la salida de features del sistema***

Cómo empezar con SRE

---

3. ***Alerting / Monitoring & Ops Load***
    - *Más logging y medición (bien)*
    - *Más alertas que llegan (probablemente mal)*
    - *Alertarse si afecta al usuario, sino apagar*
    - *Entender caminos y apps críticas*
    - *Carga operativa sólo a parte del equipo on call*
        - *Saber fácilmente cuánto tiempo está destinado a ops*
        - *Dejar al resto del equipo enfocado*

Cómo empezar con SRE

---

4. ***Blameless culture***
    - *Blame systems not humans*
    - *Modelar y hacer* ***Blameless Postmortems***

Cómo empezar con SRE

---

5. ***No intentes hacer todo a la vez***
   - 5.1. *Definir SLIs, SLOs y Error Budgets*
   - 5.2. *Auditar y ajustar monitoreo y alertas*
   - 5.3. *Modelar y compensar blameless postmortems*

Principios

service reliability engineering

- **Embracing Risks**
- **service level objectives**
- eliminating toil
- monitoring distributed systems
- automation
- release engineering
- simplicity

Embracing Risks

- *Extreme reliability comes at a cost*
  - ***Costo de HW Redundante***
  - ***Costo de oportunidad:*** *optimizando en vez de hacer otros features*
  - *Nuestro* ***objetivo*** *es* ***alinear el riesgo asumido*** *por un servicio con el riesgo que la empresa está* ***dispuesta a soportar.***
    - *Distinto para cada servicio*

Embracing Risks

SLO

- *Service Level Objectives*
- *Consideraciones*
    - *Qué nivel de servicio **espera el usuario**?*
    - *El servicio está atado al **revenue** nuestro (o de nuestros clientes?*
    - *Es un servicio **pago**, o **gratis**?*
    - *Si hay **competidores**, que SLA proveen?*
    - ***Usuarios** finales o empresas?*

Embracing Risks

Error Budget

- *Evidencia y da soporte al **conflicto** entre los objetivos de **desarrollo** y **operaciones***
- ***100%** availability es el **target equivocado** para básicamente todo*
    - *Excepción airbags :)*
- ***Product Question: the right reliability target for the system***
    - ***C**on qué **disponibilidad** estarán **conformes** los usuarios?*
    - ***Alternativas** para los que están **disconformes***
    - *Qué pasa con el uso del sistema en **distintos niveles de disponibilidad?***

Embracing Risks

Error Budget

- Error budget (unavailability) = **100% - Avail. Target**
- *El **objetivo** de **SRE***
  - *deja de ser **0 outage***
  - ***utilizar** el **error budget** para obtener la **máxima velocidad de entrega de features***
- ***Copropiedad entre SRE y Product Dev***
- ***Simplifica***
  - *decidir **cuantos releases** tendremos*
  - ***discusiones sobre outage** con stakeholders*
  - ***llegar a la misma conclusión** en distintos equipos*

Embracing Risks

Error Budget

- ***Outage*** *pasa a ser parte del proceso de innovación*

Embracing Risks

Error Budget

- ***Outage*** *pasa a ser parte del proceso de innovación*
  - *Si excedo el error budget ->*
    - ***freeze de features!***
    - ***utilizar el tiempo para ganar resiliencia***
  - *Aquí es el punto donde se pone a prueba la* ***voluntad*** *para* ***implementar SRE***

Service Level Objectives

- ***Términos***
  - SLI
    - Service Level Indicator
    - Una medición bien definida de **successful enough**
    - Usado en la definición de SLO y SLA
    - func (metric) < threshold
  - SLO
    - Service Level Objective
    - target de interacciones **exitosas**
    - lower bound ≤ **SLI** ≤ upper bound
  - SLA
    - consecuencias
    - goal
    - SLA = SLO + margen

Service Level Objectives

SLI

- ***medición cuantitativa de algún aspecto del nivel de servicio ofrecido***
  - *request latency*
  - *error rate*
  - *throughput*
- ***Algunos no podemos medirlos, y medimos un proxy***
  - ***client-side latency***
  - *medimos* ***server-side latency***
- ***Availability***

Service Level Objectives

- *Time Based Availability*

$$\text{availability} = \frac{\text{uptime}}{(\text{uptime} + \text{downtime})}$$

- *Agregate Availability*

$$\text{availability} = \frac{\text{successful requests}}{\text{total requests}}$$

Service Level Objectives

SLO

- *a **target** value or range of **values** for a **service level** that is **measured by an SLI***

Service Level Objectives

SLO

- *Estandarizar indicadores*
  - *Intervalos de agregación: “Average por minuto”*
  - *Regiones de agregación: “Todo el cluster”*
  - *Frecuencia muestreo: “cada 10 segundos”*
- *Deben especificar cómo se miden*
  - *99% (averaged over 1 minute) of Get RPC calls will complete in less than 100 ms (measured across all the backend servers).*
  - *99% of Get RPC calls will complete in less than 100 ms.*

Service Level Objectives

SLO

- *Definiendo SLOs*
  - *Percentiles?*
    - ***90%** of Get RPC calls will complete in less than 1 ms*
    - ***99%** of Get RPC calls will complete in less than 10 ms*
    - ***99.9%** of Get RPC calls will complete in less than 100 ms*
  - *No elegir un target basado en la performance actual*
  - *Mantenerlos simple*

Service Level Objectives

SLO

- *Definiendo SLOs*
  - *Evitar **absolutos***
    - *No son realistas*
      - *“Siempre...”*
      - *“Infinitamente...”*
    - *Con menos seguramente el usuario ya es feliz*
  - *Tener tan **pocos SLOs** como sea posible*
    - *Defender los **SLOs elegidos***
    - *Si no podés ganar una discusión sobre **prioridades con un SLO** en particular, no debería ser uno de los elegidos*
  - *Perfection can wait: Siempre se pueden **redefinir***

Service Level Objectives

SLA

- ***Acuerdo de Nivel de Servicio***
  - *Contrato Explícito o implícito*
  - *Incluye **consecuencias** si no se alcanza el nivel deseado*
    - *Si no las tiene, es un SLO*
    - *No siempre son económicas ni explícitas. Ej:*
      - *Google Search*
      - *Impacto indirecto:*
      - *Reputación*
      - *Ganancias por ads*
  - *Requiere **equipos legales y negocio**, para **elegir las consecuencias y penalidades adecuadas por imcumplimiento**. SRE Rol ayuda a comprender las implicancias y dificultades de cumplirlos*

Service Reliability Engineering

Otros principios de SRE

- eliminating toil
  - *“If a human operator needs to touch your system during normal operations, you have a bug. The definition of normal changes as your systems grow.”*
- monitoring distributed systems
- automation
- release engineering
- simplicity

Preguntas?

Material Extra

- https://landing.google.com/sre/books/
- Building Successful SRE in Large Enterprises (video)

Gracias!

TACS