The Rules

- 5 Minutes Per Speaker!
- 15 seconds per slide, auto advancing!
- 2 handheld mics in a Blue/Green deployment
- Speakers be ready to run on stage!

GO!!
Developing effective project plans for SRE interns

Andrew Ryan <andrewr@fb.com>
Production Engineer, Facebook Inc.
USENIX SREcon Asia, June 2019
So you’re getting interns?
What makes a good SRE intern project?

- Something experimental
  - “Rewrite an existing service in Rust”
  - If it doesn’t work, well, you still have the old one
What makes a good SRE intern project?

• Something experimental
  • “Rewrite an existing service in Rust”
  • If it doesn’t work, well, you still have the old one
• Something greenfield
  • “A new service to generate and analyze traceroute data”
  • If it doesn’t work, well, you never had it anyway 😂
Try to maximize intern interest & project impact

![Diagram showing the relationship between intern interest and project impact.]

- **Interesting, unimportant**
- **Boring, unimportant**
- **Interesting and impactful**
- **Boring, important**
Try to maximize intern interest & project impact

Make your interns ship *something*!
Combine code and operations

Operational work

Too Ops-y

Interesting, unimportant

Software engineer intern

Writing code
Combine code and operations

Give interns a real sense of SRE work

- Writing code
- Operational work
- Interesting, unimportant
- Too Ops-y
- Software engineer intern
Would *you* do this project?
#1: Title and Description

- Be clear and concise
- GOOD: “Improve reliability and speed of initial Chef runs”
- BAD: “Work on initial Chef runs”
#2: Skill requirements

- “Project requires Python, C++, and understanding of BGP”
- Great way to match interested interns with your project!
#2: Skill requirements

• This can be what interns LEARN, not just what they already KNOW

• Interns can learn QUICKLY, especially when motivated by interest!
#3: Risks

- What could go wrong?
  - External technology/team risk
  - Project more complex than you thought
#3: Risks

- What could go wrong?
  - External technology/team risk
  - Project more complex than you thought
- And how will you deal with it?
  - Alternate tech solutions
  - Change project direction/scope
#4: Milestones

- Break up work into SPECIFIC deliverables 1-2 weeks apart
- Week 2: Complete design and spec of new Foobar service
- Week 4: Have prototype built with basic functionality
#4: Milestones

- Break up work into SPECIFIC deliverables 1-2 weeks apart
- Week 2: Complete design and spec of new Foobar service
- Week 4: Have prototype built with basic functionality
- Week 6: Take 1% of production requests with new service
- Week 8: Ramp up to 20% production requests, add dashboards and monitoring
#5: Project Extensions/Minimums

- What if your intern makes more (or less) progress than anticipated?
- Or if the project turns out to be harder (or easier) than you anticipated?
#5: Project Extensions/Minimums

- What if your intern makes more (or less) progress than anticipated?
- Or if the project turns out to be harder (or easier) than you anticipated?
- Think of project extensions and independent “mini-projects”
To recap...

1. Have a written plan before the intern arrives
2. Get the plan peer-reviewed
3. Make the intern ship “something”
4. Review results, repeat next year
Thank you for helping to grow the next generation of SRE’s!
Zero Downtime cross cluster migration of microservices in Kubernetes

VMware GitHub Project: https://github.com/vmware/k8s-endpoints-sync-controller
Migration of People Vs. Migration of Services

- On planes
- Gives me anxiety

- Using our solution
- Gives me anxiety++
Maintaining SLA of services during migration is tough.
Smooth transition between two clusters with minimal risk is tougher.
The clusters can be spread across multiple regions and multiple cloud providers – it shouldn’t be a problem!
A service should be able to talk to services in other clusters in the same way that services communicate within a single cluster.
Migrating any service—whether it is stateless or stateful should happen seamlessly.
Now that you have understood our problem statement, and our obsession with memes...
How did we achieve migration of services between Kubernetes clusters?
Source Cluster

Migration Controller

VPN

Target Cluster

Migration Controller
To not replicate the service use annotation:

“vmware.com/syndicate-mode: singular”
Connecting Kubernetes Cluster

1. Clusters should have different CIDR for pods
2. All the clusters should be connected with VPN, so that there is Pod to Pod connectivity across cluster
3. The Kubernetes API Server of every cluster should be reachable to other clusters.
We have now performed k8s upgrade of our cluster using migration controller for multiple versions of k8s.
Thank You.

If you have any further questions and/or want to discuss this, please reach us at ssinghvi@vmware.com or malhotrav@vmware.com
How to ruin an SRE-Dev relationship in 3 simple steps

Raushaniya Maksudova,
Site Reliability Engineer-Software Engineer, Google
SREcon Asia 2019
Agenda

- Step#0
- Step#1
- Step#2

"WE FIXED IT. WORKS FINE IN DEV" THEY SAID

Source: https://www.uest.com/status/71790
Step#0 - [Not] Understanding

WHAT WOULD YOU SAY

YOU DO HERE?
Step#0 - [Not] Understanding

WHAT WOULD YOU SAY
YOU DO HERE?

Source: https://blog.hubspot.com/marketing/rant-content-strategists
How to fix or prevent it? Practical tips

1) Write down SRE responsibilities and share with Devs.

2) Clearly communicate that SREs and Devs have the same end goal - users’ happiness - but approach it differently.

3) Get involved in design development as early as possible.
How to fix or prevent it? Practical tips

1) Write down SRE responsibilities and share with Devs.

2) Clearly communicate that SREs and Devs have the same end goal - users’ happiness - but approach it differently.

3) Get involved in design development as early as possible.
How to fix or prevent it? Practical tips

1) Write down SRE responsibilities and share with Devs.

2) Clearly communicate that SREs and Devs have the same end goal - users’ happiness - but approach it differently.

3) Get involved in design development as early as possible.
Step#1 - [Not] Trusting

ONE DOES NOT SIMPLY JOIN THE CIRCLE OF TRUST

Source: http://www.quickmeme.com/meme/35pzpm
Step#1 - [Not] Trusting

ONE DOES NOT SIMPLY

JOIN THE CIRCLE OF TRUST

Source: http://www.quickmeme.com/meme/35pzpm
Step#1 - [Not] Trusting

ONE DOES NOT SIMPLY
JOIN THE CIRCLE OF TRUST

Source: http://www.quickmeme.com/meme/35pzpm
How to fix or prevent it? Practical tips

1) Schedule a meeting to discuss all the pain points that Devs are dealing with at the moment.

2) Ease their production pain.

3) Put Devs oncall.
How to fix or prevent it? Practical tips

1) Schedule a meeting to discuss all the pain points that Devs are dealing with at the moment.

2) Ease their production pain.

3) Put Devs oncall.
How to fix or prevent it? Practical tips

1) Schedule a meeting to discuss all the pain points that Devs are dealing with at the moment.

2) Ease their production pain.

3) Put Devs oncall.
How to fix or prevent it? Practical tips

1) Schedule a meeting to discuss all the pain points that Devs are dealing with at the moment.

2) Ease their production pain.

3) Put Devs oncall.
Step#2 - [Not] Collaborating

Source: https://imgflip.com/i/2nx0i
Step#2 - [Not] Collaborating

SAY 'COLLABORATION'

ONE MORE TIME

Source: https://imgflip.com/i/2nx0i
How to fix or prevent it? Practical tips

1) Plan things you want to work on in the next quarter together with your Dev team.

2) Set up regular (bi-weekly/monthly) meetings with Dev team to stay updated, to answer each other’s questions and offer help to each other. Over-communicate.
How to fix or prevent it? Practical tips

1) Plan things you want to work on in the next quarter together with your Dev team.

2) Set up regular (bi-weekly/monthly) meetings with Dev team to stay updated, to answer each other’s questions and offer help to each other. Over-communicate.
How to fix or prevent it? Practical tips

1) Plan things you want to work on in the next quarter together with your Dev team.

2) Set up regular (bi-weekly/monthly) meetings with Dev team to stay updated, to answer each other’s questions and offer help to each other. Over-communicate.
Revision:

- Understanding
- Trusting
- Collaborating
An Effective Agile SRE Workflow

Jay Chin
Engineering Lead - SRE
jay.chin@grab.com  @jaychin
What does an SRE do?

1. Product
2. Development
3. Capacity Planning
4. Testing + Release Procedures
5. Postmortem / RCA
6. Incident Response
7. Monitoring
8. Support Requests
9. Code Reviews
10. Etc
What does an SRE do?

1. Product
2. Development
3. Capacity Planning
4. Testing + Release Procedures
5. Postmortem / RCA
6. Incident Response
7. Monitoring
8. Support Requests
9. Code Reviews
10. Etc
Scrum ?
Kanban ? Scrumban ?
Multiple Boards for Different Workflows

1. SRE Main Project Board
2. Engineering Request Board
3. Triage Queue
Multiple Boards for Different Workflows

1. SRE Main Project Board
2. Engineering Request Board
3. Triage Queue
Multiple Boards for Different Workflows

1. SRE Main Project Board
2. Engineering Request Board
3. Triage Queue
SRE Main Projects

- This is where SRE projects go (Automation, Upgrades, Product Spikes)
- Breakdown task into small chunks (e.g. 2 days). Stuck/Complex tasks quickly become apparent in your daily standups.

Pro Tip: Limit Work In Progress (WIP)!
SRE Main Projects

• This is where SRE projects go (Automation, Upgrades, Product Spikes)

• Breakdown task into small chunks (e.g. 2 days). Stuck/Complex tasks quickly become apparent in your daily standups.

Pro Tip: Limit Work In Progress (WIP)!
Engineering Requests (Toil Board)

- Toil Work
- Categorise and Quantify Everything!
- Goal is to remove this board altogether.
Engineering Requests (Toil Board)

- Toil Work
- Categorise and Quantify Everything!
- Goal is to remove this board altogether.
Engineering Requests (Toil Board)

- Toil Work
- Categorise and Quantify Everything!
- Goal is to remove this board altogether.
Use an Interrupt Shield

1. Shield from ad-hoc requests and user support communication (online and walk-ups)
2. This permits other team members to focus on projects/automation/etc
Use an Interrupt Shield

1. Shield from ad-hoc requests and user support communication (online and walk-ups)
2. This permits other team members to focus on projects/automation/etc
Final Tips

- No one-size-fits-all solution: observe, measure, and adapt.
- Adapt Kanban columns to your workflow as you go along.
- JIRA has a print feature to use on your physical board.
Final Tips

- Online boards (JIRA or Trello) might be more suitable for remote teams.
- Set-up policies/conditions for a sticky to enter or leave a column.
- Limit number of stickies in other columns too (not just in-progress).
Thank You
jay.chin@grab.com

We are Hiring!
5 ACTIONS FOR TRAINING YOUR SRE TEAM

SRECON19 ASIA

DORIAN BASUYAU
IT ONLINE MANAGER
UBISOFT IT
UBISOFT SINGAPORE
@dorian_bas
5 ACTIONS FOR TRAINING YOUR SRE TEAM
SRECON19 ASIA

DORIAN BASUYAU
IT ONLINE MANAGER
UBISOFT IT
UBISOFT SINGAPORE
@dorian_bas
KEEP UP LEARNING!
YOUR SRE TEAM MUST LEARN TO STAY EFFICIENT
KEEP UP LEARNING!
YOUR SRE TEAM MUST LEARN TO STAY EFFICIENT
ACTION 1
HAVE A SAFE PLACE TO LEARN
ACTION 1

HAVE A SAFE PLACE TO LEARN
ACTION 2
KNOW YOUR SRE'S
ACTION 2
KNOW YOUR SRE’S
DISSOCIATE IT FROM ANY PERFORMANCE EVALUATIONS (SERIOUSLY)
DISSOCIATE IT FROM ANY PERFORMANCE EVALUATIONS (SERIOUSLY)
ACTION 3
SET YOUR TECHNOLOGY FOCUS
CHOOSE WISELY

TRAINING IS SUPPORTING YOUR STRATEGY.
CHOOSE WISELY
TRAINING IS SUPPORTING YOUR STRATEGY.
ACTION 4
CREATE TRAINING WORKGROUPS
TRAINING WORKGROUPS TO KEEP YOUR SRE’S ENGAGED
LEARNING ALONE IS NOT ALWAYS FUN
TRAINING WORKGROUPS TO KEEP YOUR SRE’S ENGAGED
LEARNING ALONE IS NOT ALWAYS FUN
ACTION 5
TRACK THE PROGRESS
SHOW THE PROGRESS TO YOUR TEAM

EVALUATE THE LEARNING PROGRESSION ON REGULAR BASIS (I.E EVERY 6 MONTHS).
SHOW THE PROGRESS TO YOUR TEAM

EVALUATE THE LEARNING PROGRESSION ON REGULAR BASIS (I.E EVERY 6 MONTHS).
THANK YOU

UBISOFT
Managing Terraform State

Mark Henderson - Stack Overflow

@thefarseeker
Terraform Module

With multiple versions

Deployed multiple times
Terraform Module

With multiple versions

Deployed multiple times
Terraform Module

With multiple versions

Deployed multiple times

Each with their own state
Terraform Module

v0.4.5

≠

State File
Terraform Module ≠ State File
Terraform Module ≠ State File
Terraform Module ≠ State File
Migration support
Migration support
Migration support
Migration support
Example Migration

```tf
module "soe" {
  source = "git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.0"
}
```

```
PS1 > C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen
[325] C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen> Update-TFModuleVersion -ToVersion 0.4.5
```

```tf
module "soe" {
  source = "git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.5"
}
```
Example Migration

```terraform
module "soe" {
  source = "git::ssh://git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.0"
}
```

PS1 > C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen

[325] C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen> Update-TFModuleVersion -ToVersion 0.4.5

```terraform
module "soe" {
  source = "git::ssh://git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.5"
}
```

@thefarseeker
Example Migration

```terraform
module "soe" {
  source = "git@ssh://git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.0"
}
```

```
PS1 > C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen

[325] C:\Users\mhenderson\Documents\Git\SRE-Ent\terraform-dev-mhen> Update-TFModuleVersion -ToVersion 0.4.5

main.tf

```terraform
module "soe" {
  source = "git@ssh://git@enterprise.ds.stackexchange.com/Ent/module.git?ref=v0.4.5"
}
```
Terraform Module

v0.4.5

State File
Thanks!
Mark Henderson
@thefarseeker
Transparency in Incident Response

How much is too much?
LIST OF PEOPLE I TRUST

WHEN THERE IS A SEV1 INCIDENT
Reliability through obscurity
Reliability through obscurity

Reliability through transparency
1 Internal transparency A (Engg team)
1. Internal transparency A (Engg team)

2. Internal transparency B (Across entire org)
1. Internal transparency A (Engg team)

2. Internal transparency B (Across entire org)

3. External Transparency A (Customers, Vendors, Stakeholders)
1. Internal transparency A (Engg team)

2. Internal transparency B (Across entire org)

3. External Transparency A (Customers, Vendors, Stakeholders)

4. External Transparency B / Global Transparency (Public)
1. Internal transparency A (Engg team)

2. Internal transparency B (Across entire org)

3. External Transparency A (Customers, Vendors, Stakeholders)

4. External Transparency B / Global Transparency (Public)
Transparency is central to SRE

Choosing SLIs, SLOs and improving upon them
Transparency is central to SRE

Choosing SLIs, SLOs and improving upon them
Transparency is central to SRE

Better understanding of SLO dependencies,
Effective policies around error budgets
Transparency is central to SRE

Better understanding of SLO dependencies,
Effective policies around error budgets
This is not decision by committee.
Explaining my SLOs be like....
Busting Myths

Twice as fast - Remove blind spots
Benefits of DevOps & SRE

- Transparency
- Ensure Balanced On-call
- Uphold & Track SLOs
- Eliminate Toil
- Simplify Incident Management
- Automate Response
- Reduce Alert Fatigue
- Increase Operational Transparency
- Conduct Blameless Postmortems
Here's a $500 discount code only for you beautiful people

"SRECONTRANSPARENCY"
Here's a $500 discount code only for you beautiful people

"SRECONTRANSPARENCY"

Meet us at Booth #6 for some awesome Swag.
“Normal” Company

Dashboard

Google

Dashboard

Dashboard

Dashboard

Dashboard

Dashboard

Dashboard

Dashboard

Dashboard
Application SLO
Service Level Objective
Standardize
Standardize
Collect Horizontally
Collect Horizontally
Service Owner create their Dashboards
Service Owner create their Dashboards
Dashboard

Application

Service

Dashboard
A = All the possible services, jobs and dashboards in your world

S = the Scope of your possible world you’re currently potentially interested in

D = the Drilldown into your dashboards, services and jobs
A = All the possible services, jobs and dashboards in your world

S = the Scope of your possible world you’re currently potentially interested in

D = the Drilldown into your dashboards, services and jobs
Thank You
@bobak
Preventative Paradigm

What SREs can learn from Moms

Rayappa Mayakunthala
@mrayappa
About Me

Director of Software Engineering
Hyderabad, India

All opinions expressed are solely my own and do not express the views or opinions of Salesforce.

Pictures are not mine and the respective owners may have copyrights.
thank you
How Moms nurture happy kids the Preventative way has some important lessons for the SREs.

We can build, scale and run healthy distributed systems by leveraging the same principles.
Preventative by design - Nurturing Happy Kids
Reliability

- Effective Monitoring
- Being on-call
- Tracking outages
Reliability

- Effective Monitoring
- Being on-call
- Tracking outages
RELIABILITY

- Effective Monitoring
- Being on-call
- Tracking outages
Security

- Embrace Risk
- Address cascading failures
- Integrity
Security

- Embrace Risk
- Address cascading failures
- Integrity
Security

- Embrace Risk
- Address cascading failures
- Integrity
Scalability

- Pipelines
- Blameless RCA
- Config Management
Scalability

- Pipelines
- Blameless RCA
- Config Management
Scalability

- Pipelines
- Blameless RCA
- Config Management
Performance

- SLO’s
- Manage Load
- Eliminate Toil
- Learn/Teach/Mentor
Performance

- SLO’s
- Manage Load
- Eliminate Toil
- Learn/Teach/Mentor
Performance

- SLO’s
- Manage Load
- Eliminate Toil
- Learn/Teach/Mentor
Preventative by design - Nurturing Happy Kids
Preventative by design - Nurturing Happy Kids
SRE needs more women engineers
Our Practices of Delegating Ownership in Microservices World

Daisuke Fujita (@dtan4)
Mercari, Inc.
Microservices Platform Team

June 14, 2019
SREcon19 Asia/Pacific Lightning Talk
mercari
delegating ownership?

At Mercari Microservices
Monolith (-2018)

Good at first, but **not scalable**

- ▲ Complexity
- ▼ Velocity

Need to change the system & organization, **before** growing up more
Monolith (-2018)

Good at first, but **not scalable**

- ▲ Complexity
- ▼ Velocity

Need to change the system & organization, **before** growing up more
Microservices at Mercari

Scale the organization & Maximize output

👉 small / autonomous / cross-functional teams

👉 strong ownership by the teams
Microservices at Mercari

Scale the organization & Maximize output

👉 small / autonomous / cross-functional teams

👉 strong ownership by the teams
Service teams write code, deploy it, run it by themselves
Ownership by Microservice Team

Service teams write code, deploy it, run it by themselves

...how? 🤔
Q: How to prepare microservice’s infrastructure?
Monorepo for Terraform Configurations
Monorepo for Terraform Configurations
Monorepo for Terraform Configurations

terraform/microservices
|-- mercari-echo-jp
|  |-- development
|  `-- production
|     |-- backend.tf
|     |-- google_bigquery_dataset.tf
|     |-- google_spanner_database.tf
|     |-- module_microservice_starter_kit.tf
|     |-- providers.tf
|     `-- variables.tf
`-- mercari-listing-jp
  |-- development
  `-- production

Reviewers

mercari/mercari-echo-jp-prod is a code owner

mercari-echo-jp...

At least 1 approving review is required to merge this pull request.
Q: What are the microservices requirements in production?
Production Readiness Checklist

This documentation describes a checklist. You can use the checklist to check if a microservice is production ready or not. A microservice team must meet this requirements before production rollout. As we understand no one-size-fits-all, we defined some levels in the readiness.

- 🌟 **Level A** - for a **critical** microservice
- ✡ **Level B** - for a **standard** microservice
- ⚡ **Level C** - for an **experimental** microservice

See [Production Readiness Level](#) for details.
Production Readiness Checklist

This documentation describes a checklist. You can use the checklist to check if a microservice is production ready or not. A microservice team must meet this requirements before production rollout. As we understand no one-size-fits-all, we defined some levels in the readiness.

- 🌟 Level A - for a critical microservice
- ⭐ Level B - for a standard microservice
- ⚡ Level C - for an experimental microservice

See Production Readiness Level for details.
Production Readiness Checklist

- Maintainability
- Durability
- Observability
- Reliability
- Security
- Accessibility
- Sustainability
- Data Storage

Reliability

Reliability affects availability and productivity. If reliability is low, your system will break down often. The team will have to take time to fix it. Then, both system availability and team productivity will be decreased.

<table>
<thead>
<tr>
<th>Check name</th>
<th>Short Description</th>
<th>Level C</th>
<th>Level B</th>
<th>Level A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Scale</td>
<td>It can be manually scaled horizontally.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Auto Scale</td>
<td>It automatically scales horizontally.</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CPU req/limit</td>
<td>Its CPU usual usage is 90% of CPU resource request value, and 40% of CPU resource limit value.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Memory req/limit</td>
<td>Its memory resource request value is as same as limit value.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Capacity planning</td>
<td>Its capacity is clear by doing load test</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Zero downtime</td>
<td>Its deploy process does not cause service degradation or downtime (e.g. error rate does not increase)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Production Readiness Checklist

TODO: automated regular check

GitHub issue-based check & review process

[mercari-echo-jp] Production Readiness Review (Level B)
#2394

TODO: automated regular check
Production Readiness Checklist

GitHub issue-based check & review process

[mercari-echo-jp] Production Readiness Review (Level B)
#2394

TODO: automated regular check
Conclusion

Ownership by microservice teams

- Monorepo for provisioning microservice infrastructure
- Production Readiness Checklist

It contributes to organizational expansion with microservices

https://careers.mercari.com
Conclusion

Ownership by microservice teams

- Monorepo for provisioning microservice infrastructure
- Production Readiness Checklist

It contributes to organizational expansion with microservices

https://careers.mercari.com