



salesforce

# Engineering empathy

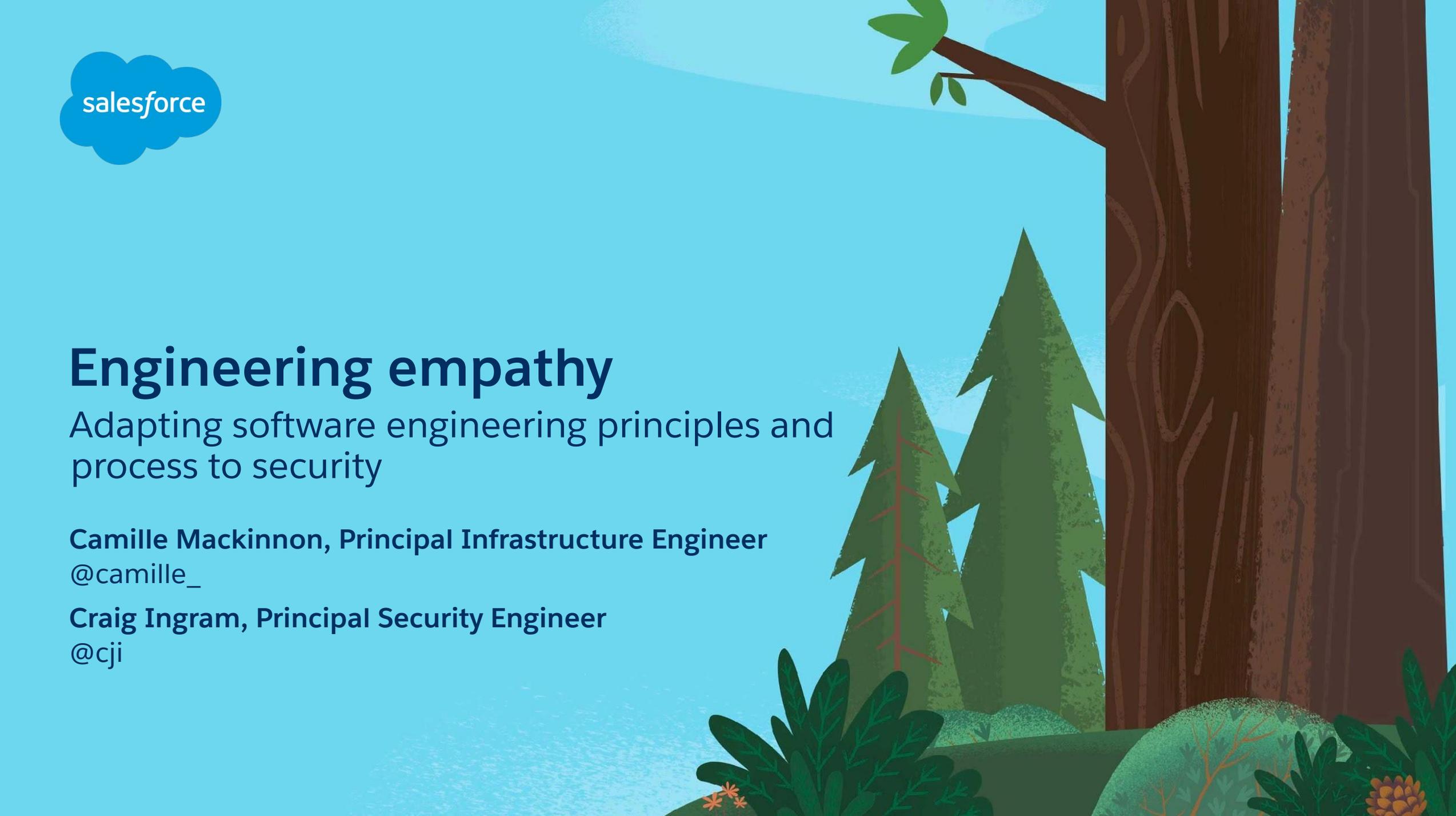
Adapting software engineering principles and process to security

**Camille Mackinnon, Principal Infrastructure Engineer**

@camille\_

**Craig Ingram, Principal Security Engineer**

@cji



# Who are we?



**Camille Mackinnon**  
Principal Infrastructure engineer



**Craig Ingram**  
Principal Security engineer



# Learning from software engineers

salesforce

Everyday principles behind writing good code

How software engineers plan their work and processes

Devops culture



# Learning from software engineers

salesforce

Everyday principles behind writing good code

How software engineers plan their work and processes

Devops culture



# Software design principles



KISS (Keep it Small and Simple)

DRY (Don't Repeat Yourself)

TDD (Test-Driven Development)



# KISS (Keep it Small and Simple)



Keep recommendations simple and straightforward

Try giving your engineering partners actionable clarity



# KISS (Keep it Small and Simple)



Not just simple, but also as easy as possible to execute

Keeping things simple for the engineers may mean more work for you in the short term



# DRY (Don't Repeat Yourself)

Reduce repetition of software patterns



Modification of one element does not require a change in other unrelated elements



# DRY (Don't Repeat Yourself)



Similar findings, in multiple places in a code base

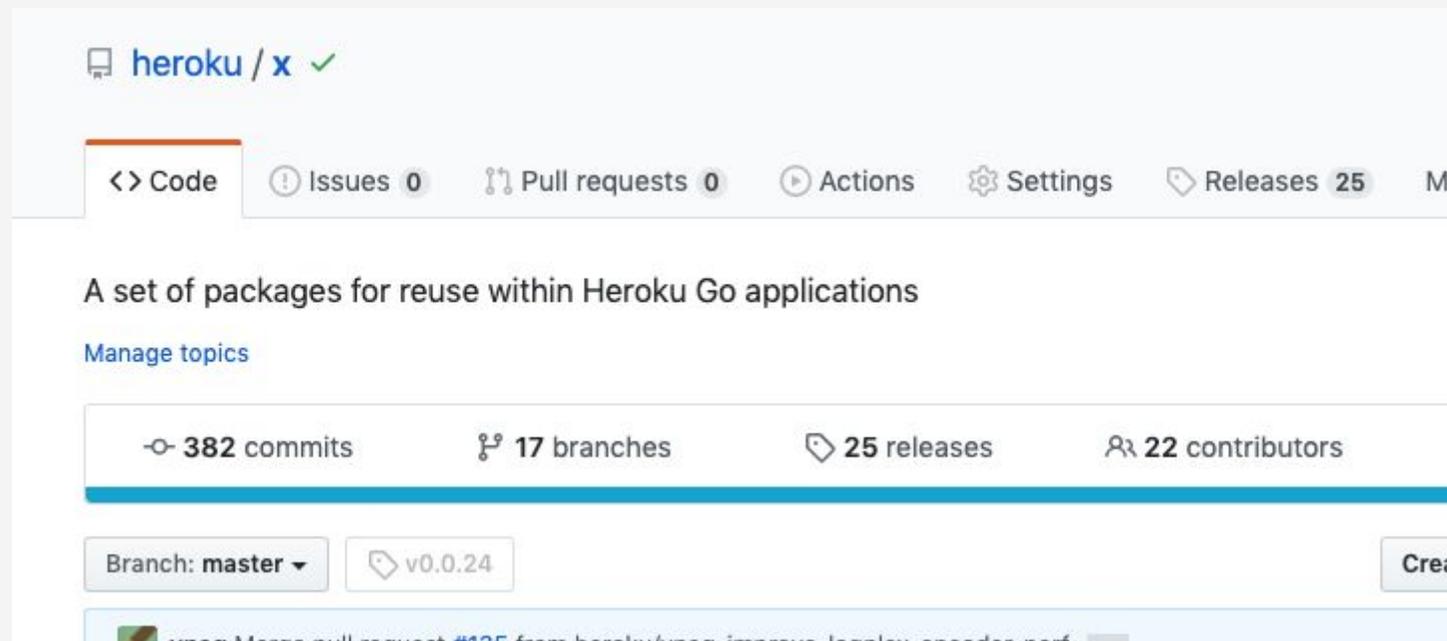
The same security findings, again and again - why?



# DRY (Don't Repeat Yourself)



Give engineering teams the correct libraries and abstractions to build with



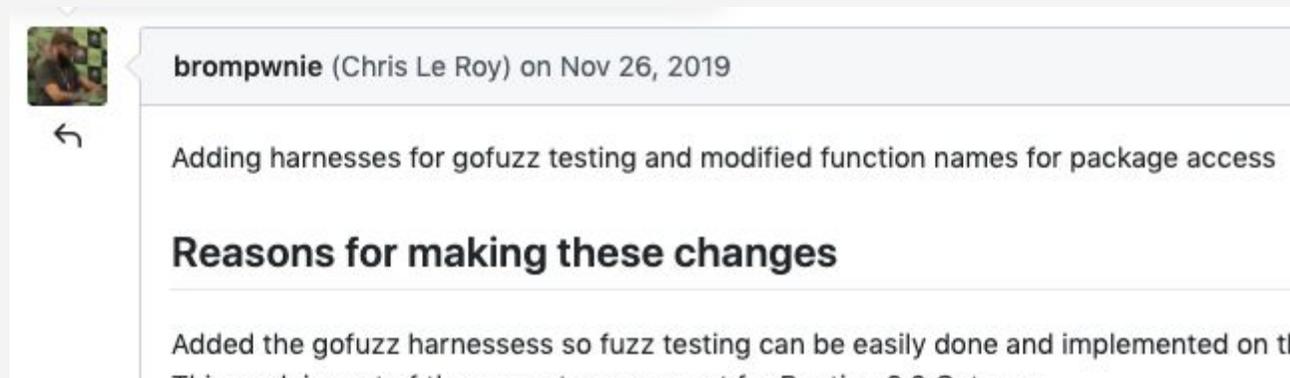
# TDD (Test-Driven Development)

Start with failure

Prevent regressions by putting automated tests in place with patches

```
// security check for bugs #XXX and #XXX
const afterEncrypted = await Project.findById(project.id).encrypted_access_token
expect(beforeEncrypted).to.not.equal(afterEncrypted)
})
})
```

Integrate fuzz testing with a project that re-finds your bug(s)



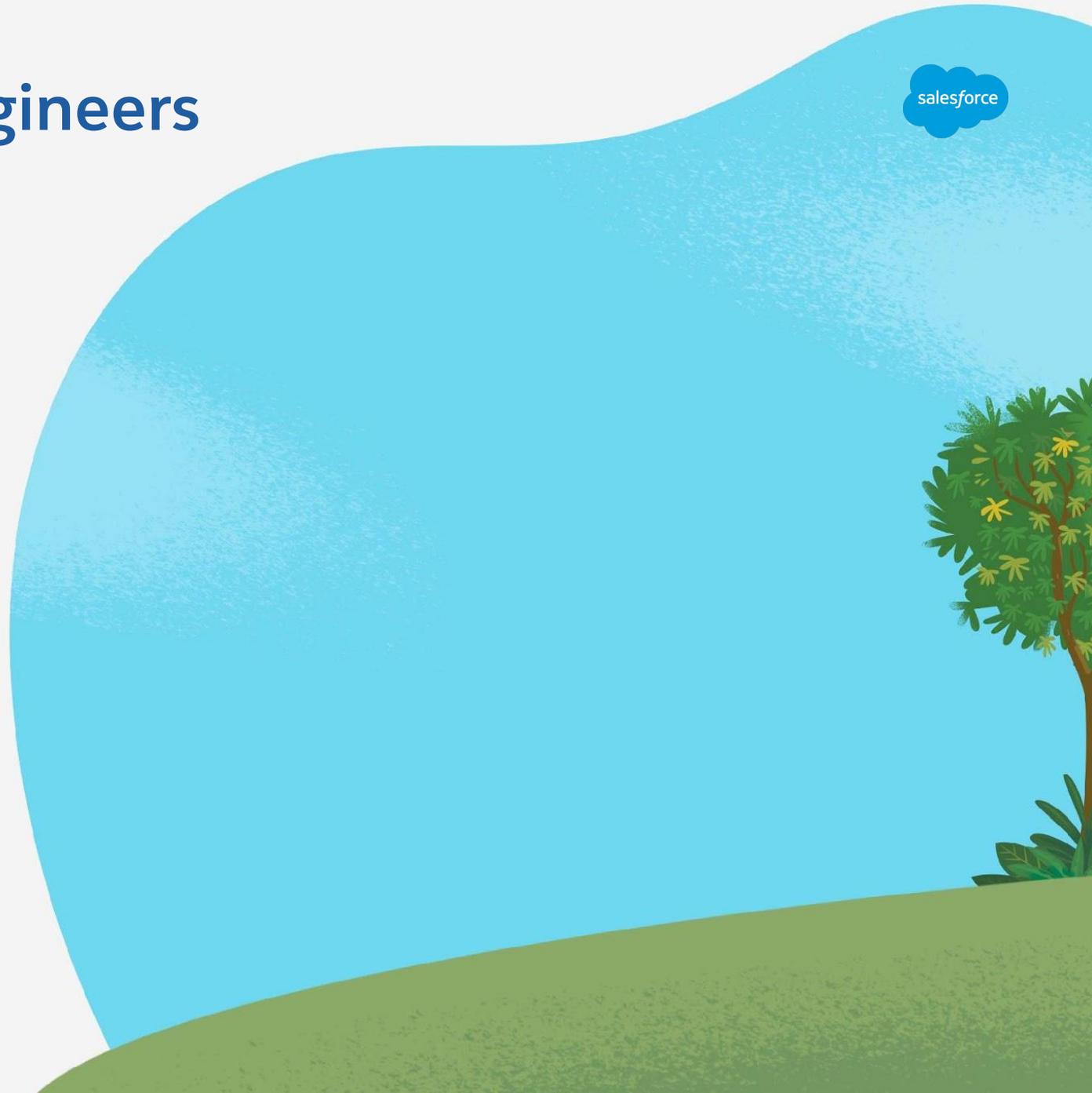
# Learning from software engineers

salesforce

Everyday principles behind writing good code

How software engineers plan their work and processes

Devops culture



# Software engineering processes



Prioritization and planning

Retrospectives

User research



# Prioritization, work and capacity planning



Engineering teams need to prioritize amid many competing aspects of their work:

- new features customers need
- performance and scalability
- security!
- long term architectural changes



# Prioritization, work and capacity planning



Balance:

- security assessments (audit deadlines, engineering deadline)
- tool development
- deeper research



# Prioritization, work and capacity planning



Ruthless prioritization

Clear objectives and key results (OKRs)

## Objective

Reduce our supply chain security exposure by integrating a scanning tool that checks all pull requests for vulnerable software dependencies.

## Key Results

- Deployment covers 100% of source code repositories for production components
- Self-service ability for engineering teams to continue to onboard new components
- Remediation of existing Critical/High severity findings in 3 months
- Ongoing remediation of new findings within company SLAs



# Retrospectives



Engineering teams run retrospectives for:

- their sprints
- major downtime incidents

Blame-free, identify steps forward



# Retrospectives



Even without sprints, taking time to identify as a team if we are making the progress we want is important

Security incidents, major bugs found after release, security flaws in architecture can all benefit from a retrospective



# Customer interviews, user research



Engineering teams rely on product managers, user researchers or themselves to figure out if they are building the **right** thing. They:

- do customers interviews or user research
- develop user personas to understand their users' needs



# Customer interviews, user research



Listen to your engineers and understand what they are asking

Observe what your engineers are doing and what that can tell you about their needs



# Customer interviews, user research



Metro Centric / CC BY  
(<https://creativecommons.org/licenses/by/2.0>)



# Customer interviews, user research



An engineer trying to work around security controls can tell you a lot about solutions you need to provide them.

Understanding your engineering team's needs means that recommendations are more likely to be followed



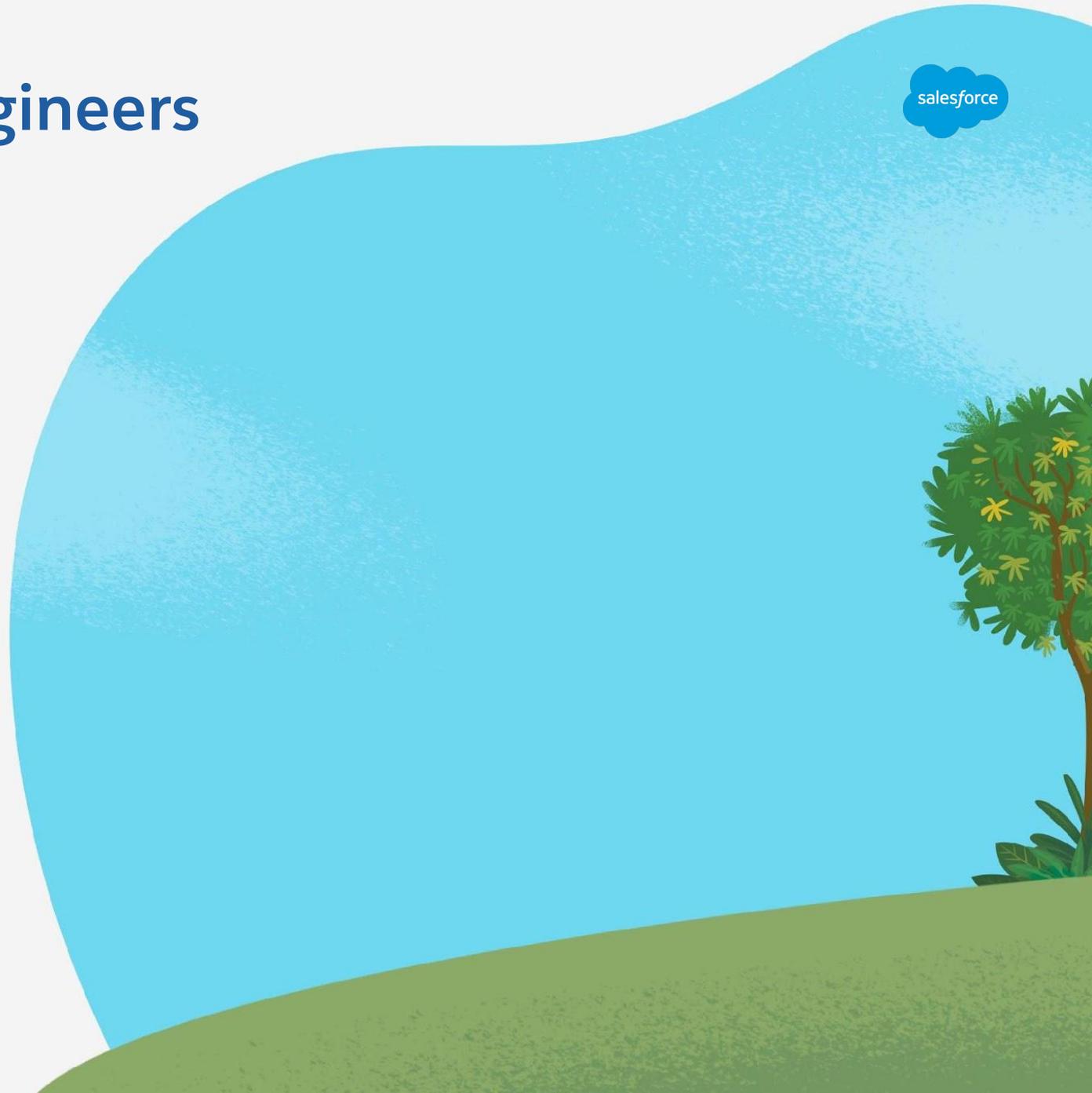
# Learning from software engineers

salesforce

Everyday principles behind writing good code

How software engineers plan their work and processes

**Devops culture and organizational changes**



# Devops culture and Organizational level changes



Measures: SLOs, SLIs

Remove toil through automation

Shifting left



# Measures: SLOs, SLIs



Service level objectives and indicators

Actions are taken when indicators reach certain levels



# Measures: SLOs, SLIs



Service level indicators for infosec:

- last time each team has asked for a security review
- last time your threat models have been updated
- pull request failure rates due to automated security tools



# Remove toil through automation



Toil: manual, repetitive, automatable, reactive, lacks enduring value, grows at least as fast as its source

Integrate with engineers' own efforts to automate: see what tools they are using and how to leverage them to automate security work as well



# Shifting left



Be a part of the early planning and design decisions

Adapt to the environment you are in

Build tools and automation that works with what teams are already using



# Conclusion



Look at your software engineering teams to improve your security team

Develop empathy by understanding their constraints and competing priorities

Share in their culture to work more closely together and provide better security solutions



# References



- <https://www.blackhat.com/us-19/briefings/schedule/index.html#every-security-team-is-a-software-team-now-17280>
- <https://twitter.com/brompwnie>
- <https://github.com/koenbuyens>
- <https://twitter.com/kmcquade3>
- [https://en.wikipedia.org/wiki/KISS\\_principle](https://en.wikipedia.org/wiki/KISS_principle)
- [https://en.wikipedia.org/wiki/Don%27t\\_repeat\\_yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself)
- <https://rework.withgoogle.com/guides/set-goals-with-okrs/steps/introduction/>
- <https://codeascraft.com/2012/05/22/blameless-postmortems/>
- <https://landing.google.com/sre/sre-book/chapters/service-level-objectives/>



Thank You

