








## Module 6: Apply SRE in Your Organization

Throughout the last several modules, you've learned about Google's SRE practical and cultural fundamentals and ways you can adopt them in your own organization. Before you can start employing many of the technical principles, it's important to understand a few additional points.

# Learning topics



---

-  Assess organizational maturity level for SRE
-  Necessary SRE skills
-  How to train your workforce
-  SRE team implementations
-  How Google Cloud can support your organization

In this module, we'll discuss how you can assess and understand your organization's maturity and readiness for adopting SRE principles, practices, and culture.

You'll also learn what skills are necessary in an SRE and how to train your current workforce.

Lastly, we'll provide examples of SRE team implementations and the additional support our Google Cloud Professional Services teams can provide as you continue on your journey to SRE.

 SLOs with Consequences Make Tomorrow Better  
than Today Regulate Workload

It's important that you assess your organization's maturity level for adopting SRE before you implement the various principles.

We've looked at Site Reliability Engineering as a three part journey: **SLOs with Consequences, Make Tomorrow Better than Today, and Regulate Workload.**

## Organizational maturity for SRE

**Low:** No adopted SRE principles, practices, or culture

**High:** Well-established SRE team or widely embraced principles, practices, and culture

Organizational maturity is considered **low** if your organization has not yet adopted SRE principles, practices, and culture. Organizational maturity is considered **high** if you have a well-established SRE team, or if SRE principles, practices, and culture are widely understood, implemented, and embraced.

## High SRE maturity

- Well-documented and user-centric SLOs
- Error budgets
- Blameless postmortem culture
- Low tolerance for toil

An organization with high SRE maturity is expected to have the following:

- Well-documented and user-centric **SLOs**, where a target level of reliability is ideally correlated with customer happiness.
- **Error budgets**, which are budgets for failure. The error budget is the difference between perfection and your SLO. They allow IT teams to move fast, as long as the budget is not exhausted, but with defined actions to improve reliability if the production service fails.
- **A blameless postmortem culture**. This culture recognizes that things will go wrong and that human errors are really systems problems.
- **A low tolerance for toil**. Again, toil is work that tends to be manual, repetitive, automatable, tactical, and devoid of enduring value, and that scales linearly as a service grows.

These principles can be adopted by any team responsible for production systems—regardless of its name—before and in parallel to staffing an SRE team. As you can see, these are many of the technical principles we’ve discussed throughout the course.



At Google, we believe we can't achieve the business goals mentioned earlier—such as aligning development and operations incentives, balancing feature velocity with reliability, and fostering effective collaboration—without first defining SLOs and error budgets and setting policies around them. It's possible that your journey to SRE may have a different first step based on your organizational culture and business needs. However these are the Google-recommended first steps.

After setting up your SLOs, error budgets, and postmortem culture, you will be better prepared to start defining and organizing your SRE teams. After that, you can start implementing practices for automation and regulating workload.

## DORA DevOps Quick Check tool

<https://www.devops-research.com/quickcheck.html>

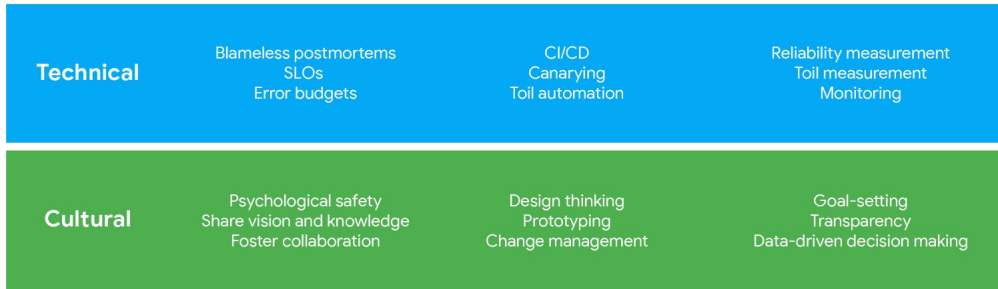
You can also use this tool, called the DORA DevOps Quick Check, for a recommendation on how to get started. It's a quick, five-question assessment about your current engineering practices. Access the tool at <https://www.devops-research.com/quickcheck.html>.



1

2

3



Work with your engineering and operations teams to understand where your organization fits into this journey. Your organization might not use any SRE principles yet. And that's okay! Hopefully now you have a clearer understanding of the principles to start with. We at Google want to help you get started. We'll talk about how this works at the end of this module.





In the next video, we'll talk about the types of skills Site Reliability Engineers need to have and develop to make your SRE team productive and successful.



In addition to understanding the practical and cultural fundamentals that we've discussed so far in this course, SREs should have several skills to support your organization's adoption of Site Reliability Engineering.

## Who to hire

- Engineers with SRE experience
- Systems administrators with operations and scripting experience

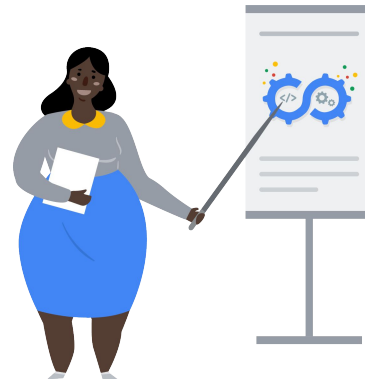


### Who to hire

As you've learned, an SRE is an engineer who also runs operations. Unless you're hiring an **engineer who has already worked as an SRE**, you'll likely need to upskill any current or new engineers you put into the role. SRE concepts are not often taught in school or university. At Google, we also hire SREs that were **systems administrators who have worked operations and who also have some scripting experience**. We offer them software engineering training to improve their engineering skills.

Since you're making a large organizational shift to SRE, it's important to understand that you'll need to provide necessary training and resources for people moving into SRE roles. It's an investment that will benefit your organization in the long run.

## What skills to train and hire



### **What skills to train and hire**

At Google, we've discovered that there are particular skills to focus on when training and hiring new SREs. This is by no means a comprehensive list, but it's what Google recommends as essentials when getting started.

## What skills to train and hire

- **Operations and software engineering**



### ***Operations and software engineering***

Running applications in production gives invaluable insights that cannot be easily gained otherwise, so engineers with little or no experience in operations will need to be upskilled. Additionally, systems administrators or other operators can also make great SREs, so they'll need to understand the software they are supporting and be empowered to improve it.

## What skills to train and hire

- Operations and software engineering
- **Monitoring systems**

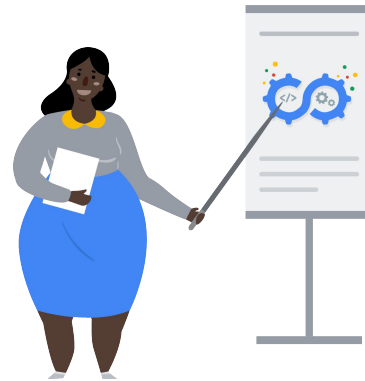


### ***Monitoring systems***

SRE principles require SLOs that can be measured and accounted for, so an understanding of monitoring systems is necessary.

## What skills to train and hire

- Operations and software engineering
- Monitoring systems
- **Production automation**

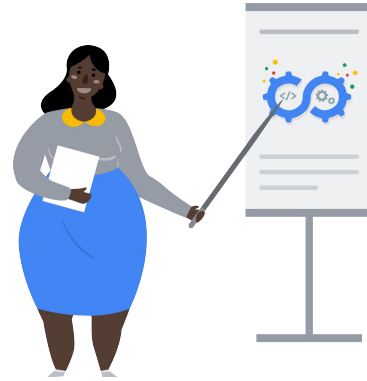


### ***Production automation***

Scaling operations requires automation. Your SREs will need an understanding of how to automate processes.

## What skills to train and hire

- Operations and software engineering
- Monitoring systems
- Production automation
- **System architecture**



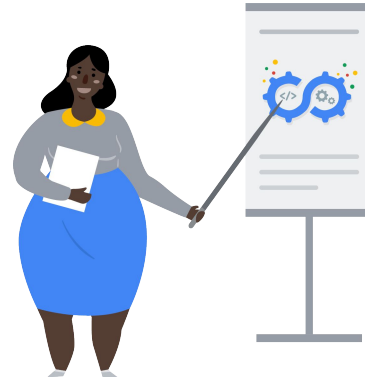
### ***System architecture***

Scaling an application requires good architecture, so SREs should have a skill set that includes understanding system architecture and how to work with and create it.



## What skills to train and hire

- Operations and software engineering
- Monitoring systems
- Production automation
- System architecture
- **Troubleshooting**

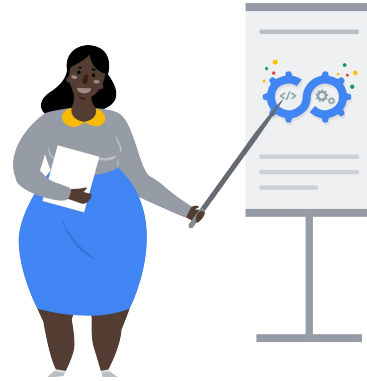


### ***Troubleshooting***

SREs are regularly on-call and therefore require sharp troubleshooting skills. They should be inquisitive and follow an analytical approach to solving problems.

## What skills to train and hire

- Operations and software engineering
- Monitoring systems
- Production automation
- System architecture
- Troubleshooting
- **Culture of trust**

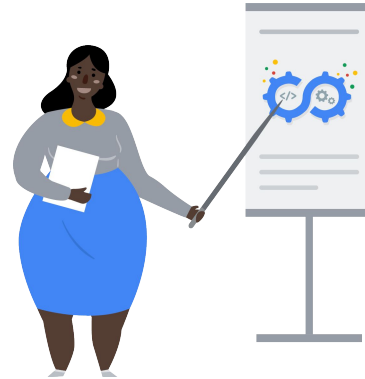


### ***Culture of trust***

Because SREs and developers share ownership of services and what customers experience, they need to have a good relationship. At Google, we've learned that the top three characteristics of a healthy culture between SREs and developers are communication, agreement, and trust. It's important to extend this culture to other teams as well, such as security and privacy teams.

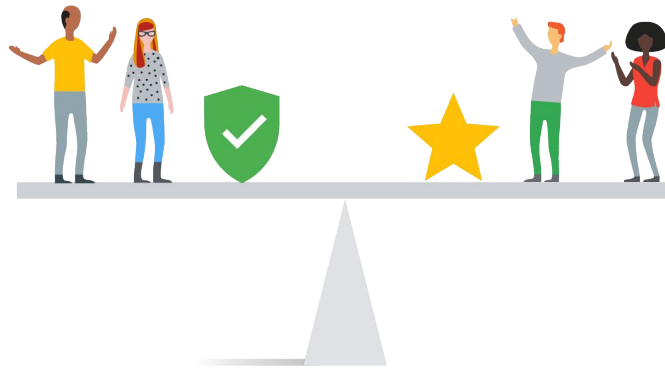
## What skills to train and hire

- Operations and software engineering
- Monitoring systems
- Production automation
- System architecture
- Troubleshooting
- Culture of trust
- **Incident management**



### ***Incident management***

Incident management has two sides: technical troubleshooting and communication framework. For timely resolution for incidents, SREs will need to both technically resolve problems and interact with many other people. They should be adept in explicit and clear communication, time and task management, and record keeping, to name a few skills.



Look for **resilience** and **flexibility** in an SRE.

As you build out your SRE team, you should remember that these individuals will be in a difficult and ambiguous position between feature velocity and reliability goals. Important character traits to look for are **resilience and flexibility**, because SREs will need to provide the right balance between enabling product development and doing what's right for your customers.



If you start with these skills and begin adopting the practices of SLOs, error budgets, and postmortems, you'll be well-prepared to start implementing your first SRE team.

In the next video, we'll give you an overview of several different types of SRE teams that you can consider for your organization.



Once you have established some key practices and have begun training and hiring for the SRE role in your organization, you can start thinking about how to implement your SRE team. This implementation can and will look different depending on the size of your organization and where you are on your journey to SRE.

In this video, we'll give you an overview of the different types of implementations and the benefits and disadvantages of each.

## Google-recommended SRE team implementations

1. Kitchen Sink, or “Everything SRE”
2. Infrastructure
3. Tools
4. Product/Application
5. Embedded
6. Consulting

We've categorized our recommended implementations into six different categories:

1. Kitchen Sink, or “Everything SRE”
2. Infrastructure
3. Tools
4. Product/Application
5. Embedded
6. And Consulting

## Kitchen Sink/“Everything SRE” team

- Its scope is unbounded.
- It is a good starting point for first SRE team.
- It is recommended for organizations with few applications and user journeys.
- It is useful when a dedicated SRE team is needed.

The first is **Kitchen Sink** or “**Everything SRE.**”

Scope is usually unbounded with this type of team. If you’ve never created an SRE team, this is a good place to start. We recommend this approach for organizations that have few applications and user journeys, where the scope is small enough that only one team is necessary, but a dedicated SRE team is needed in order to implement its practices.



## Benefits of Kitchen Sink/“Everything SRE”

- There are no coverage gaps.
- It is easy to spot patterns and similarities between services and projects.
- It acts as glue between teams.



There are several benefits to the Kitchen Sink implementation:

- There are no coverage gaps between SRE teams, given that only one team is in place.
- It's easy to spot patterns and draw similarities between services and projects.
- SRE tends to act as a glue between disparate developer teams, creating solutions out of distinct pieces of software.

## Disadvantages of Kitchen Sink/“Everything SRE”

- It usually lacks a team charter.
- It risks overloading the team.
- It can run the risk of shallow contributions.
- Team issues can have a negative impact on the business.



There are also several disadvantages:

- There is usually a lack of an SRE team charter, or the charter states that everything in the company can be in scope, running the risk of overloading the team.
- As the company and system complexity grows, the team tends to move from having a deep, positive impact on everything to making many more shallow contributions.
- Issues involving the team may negatively impact your entire business.

## Infrastructure team

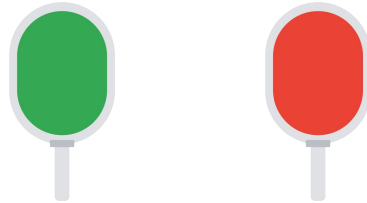
- It helps make other team's jobs easier.
- It maintains shared services related to infrastructure.
- It is recommended for organizations with multiple developer teams.
- It defines common standards for the IT team.

The second implementation is **Infrastructure**.

This type of team focuses on behind the scenes tasks that help make other teams' jobs easier and faster. They work on maintaining shared services and components related to infrastructure, versus an SRE team dedicated to working on services related to products, like customer-facing code. We recommend this implementation for a company with several development teams, since they probably need to staff an infrastructure team to define common standards and practices. It is common for large companies to have both an infrastructure DevOps team and an SRE team, where the DevOps team focuses on features and SRE focuses on reliability.

## Benefits of Infrastructure

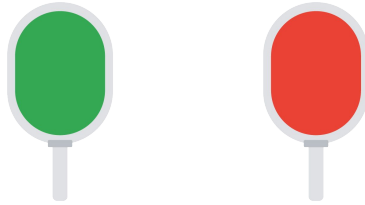
- It allows developers to use DevOps practices without divergence across business.
- It keeps its focus on highly reliable infrastructure.
- It defines production standards.



A benefit of the Infrastructure implementation is that it allows product developers to use DevOps practices to maintain user-facing products without divergence in practice across the business. Additionally, SREs can focus on providing a highly reliable infrastructure. They will often define production standards as code and work to smooth out any sharp edges to greatly simplify things for the product developers running their own services.

## Disadvantages of Infrastructure

- It has possible negative impact to business following team issues.
- Improvements the team makes may not be tied to customer experience.
- It may require teams to be split, which can lead to duplication or divergence of practices.



However there are some disadvantages:

- Depending on the scope of the infrastructure, issues involving such a team may negatively impact your entire business, similar to a “Everything SRE” implementation.
- Lack of direct contact with your company’s customers can lead to a focus on infrastructure improvements that are not necessarily tied to the customer experience.
- As the company and system complexity grows, you may be required to split the infrastructure teams, which can lead to duplication of base infrastructure or divergence of practices between teams, which is inefficient and limits knowledge sharing and mobility.

## Tools team

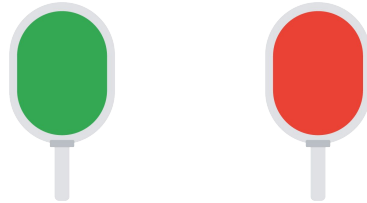
- It focuses on building software to help developers with aspects of SRE work.
- It is recommended for organizations that need highly specialized reliability-related tooling.

Third is a **Tools**-focused SRE team.

This type of SRE team tends to focus on building software to help their developer counterparts measure, maintain, and improve system reliability or other aspects of SRE work, such as capacity planning. A tooling SRE team risks solving the wrong problems for the business, so it needs to stay aware of the practical problems front-line reliability teams are addressing. This kind of team is recommended for any organization that needs a highly specialized reliability-related tooling.

## Benefits of Tools

- It allows developers to use DevOps practices without divergence across business.
- It keeps its focus on highly specialized reliability-related tooling.
- It defines production standards.



## Disadvantages of Tools

- It could unintentionally turn into an infrastructure team.
- There is risk of increased toil and overall workload on the team.

The benefits and disadvantages of infrastructure and tools SRE teams tend to be similar. Additional disadvantages for tools teams include:

- Ensuring that the team doesn't unintentionally turn into an infrastructure team, and vice versa.
- Running the risk of an increase of toil and overall workload. This is usually contained by establishing a team charter that business leaders approve.

## Product/Application team

- It improves the reliability of a critical application.
- It is recommended for organizations that have Kitchen Sink, Infrastructure, or Tools SRE team and application with high reliability needs.

The next implementation is **Product/Application**.

This kind of SRE team works to improve the reliability of a critical application or business area. We recommend this implementation for organizations that already have a Kitchen Sink, Infrastructure, or Tools-focused SRE team and have a key user-facing application with high reliability needs. Having each of these aspects justifies the relatively large expense of a dedicated set of SREs.

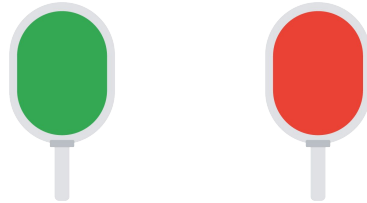


### Benefits of Product/Application

- It provides clear focus.
- It creates a clear link between business priorities and team effort expenditure.

### Disadvantages of Product/Application

- It may require establishing new teams as the business and complexity grow.
- It can lead to duplication of infrastructure and divergence of practices.



The benefit of this approach is that it provides a clear focus for the team's effort and creates a clear link between business priorities and where the team spends effort.

The disadvantage is that, as the company and system complexity grow, the organization will require new product/application teams. As with an infrastructure team, the focus of each team can lead to duplication of base infrastructure or divergence of practices, which is inefficient and limits knowledge sharing and mobility.

## Embedded team

- SREs are embedded with developers.
- SREs and developers have a project- or time-bounded relationship.
- It is hands-on, changing code and configuration of services.
- It is recommended for organizations to start a team or scale another implementation.
- It can augment the impact of a tools or infrastructure team.

The fifth type of SRE team implementation is an **Embedded** team.

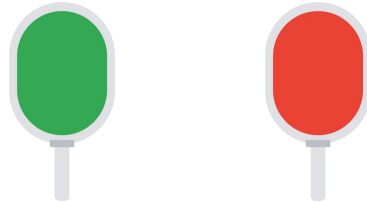
This team has SREs embedded with their developer counterparts, usually one per developer team in scope. An embedded SRE usually shares an office with the developer, but the embedded arrangement can be remote. The work relationship between the embedded SREs and developers tends to be project- or time-bounded. During embedded engagements, the SREs are usually very hands-on, performing work like changing code and configuration of the services in scope. We recommend this implementation to either start an SRE function or scale another implementation. This model is useful when you have a project or team that needs SRE for a period of time. This type of team can also augment the impact of a tools or infrastructure team by driving adoption.

### Benefits of Embedded

- It is focused expertise directed to specific problems or teams.
- It allows side-by-side demonstration of SRE practices.

### Disadvantages of Embedded

- It can cause a lack of standardization between teams.
- It can lead to divergence in practice.
- There is less time for mentoring.



A benefit of an embedded SRE team is that focused SRE expertise can be directed to specific problems or teams. It also allows side-by-side demonstration of SRE practices, which can be a very effective teaching method.

Disadvantages are that it may result in lack of standardization between teams, or divergence in practice, and SREs may not have the chance to spend much time with peers to mentor them.

## Consulting team

- It is similar to an Embedded team.
- It is less hands-on.
- SREs may write code and maintain tools for themselves and developers.
- It is not recommended until organizational complexity is large.
- Google recommends staffing one to two part-time consulting SREs before the first SRE team.

Our last recommended SRE implementation is **Consulting**.

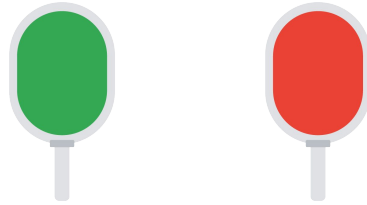
This implementation is very similar to the embedded implementation. The difference is that consulting SREs are less hands-on. These SREs tend to avoid changing customer code and configuration of the services in scope. However, they may write code and configuration to build and maintain tools for themselves or for their developer counterparts, which is a hybrid of the consulting and tools-focused SRE role. We recommend waiting to staff a dedicated SRE team of consultants until your organization or complexity is considered to be large, and when demands have outgrown what can be supported by existing SRE team implementations. We also recommend staffing one or two part-time consultants before you staff your first SRE team.

## Benefits of Consulting

- It can help with the scaling of an existing SRE team's positive impact.
- It is decoupled from directly changing code and configuration.

## Disadvantages of Consulting

- It may lack sufficient context to offer useful advice.
- It can be perceived as hands-off.



The benefit of a consulting SRE team is that it can help with additional scaling of an existing SRE team's positive impact by being decoupled from directly changing code and configuration.

The disadvantage is that consultants may lack sufficient context to offer useful advice. Additionally, a common risk for consulting SRE teams is being perceived as hands-off, given that they typically don't change code and configuration, even though they are capable of having indirect technical impact.



Do one or any of these SRE team implementations resonate or appeal to you for your organization? Remember that it's important to assess your organization's maturity for SRE adoption and identify areas for training before creating your first SRE team.

In the next and final video, you'll learn how Google can help your organization get started with SRE.



So, you see the value of implementing SRE practices and teams in your organization. What now? Google Cloud is here to support you!

We want to make sure we provide valuable experience to our customers. Google Cloud's Professional Services team specializes in supporting organizations like yours in jumpstarting its SRE journey.

Reach out to your Account Director or  
Account Executive to request  
a Google Professional Services consultation.

To request Google Cloud Professional Services consultation, be sure to reach out to your [role] to start the conversation.





Thank you so much for coming along this SRE journey with us! We hope you've gained valuable insights into Google SRE culture and how it can benefit your IT operations and your business.

To make sure you've fully grasped the SRE concepts we've covered, and to get credit for this course, complete the final assessment in the last module.

Finally, please be sure to review the Resources section at the end of this course for additional materials on Google SRE. You can also find these resources in your Learner Workbook.

Good luck on your SRE journey! We hope to hear from you soon.