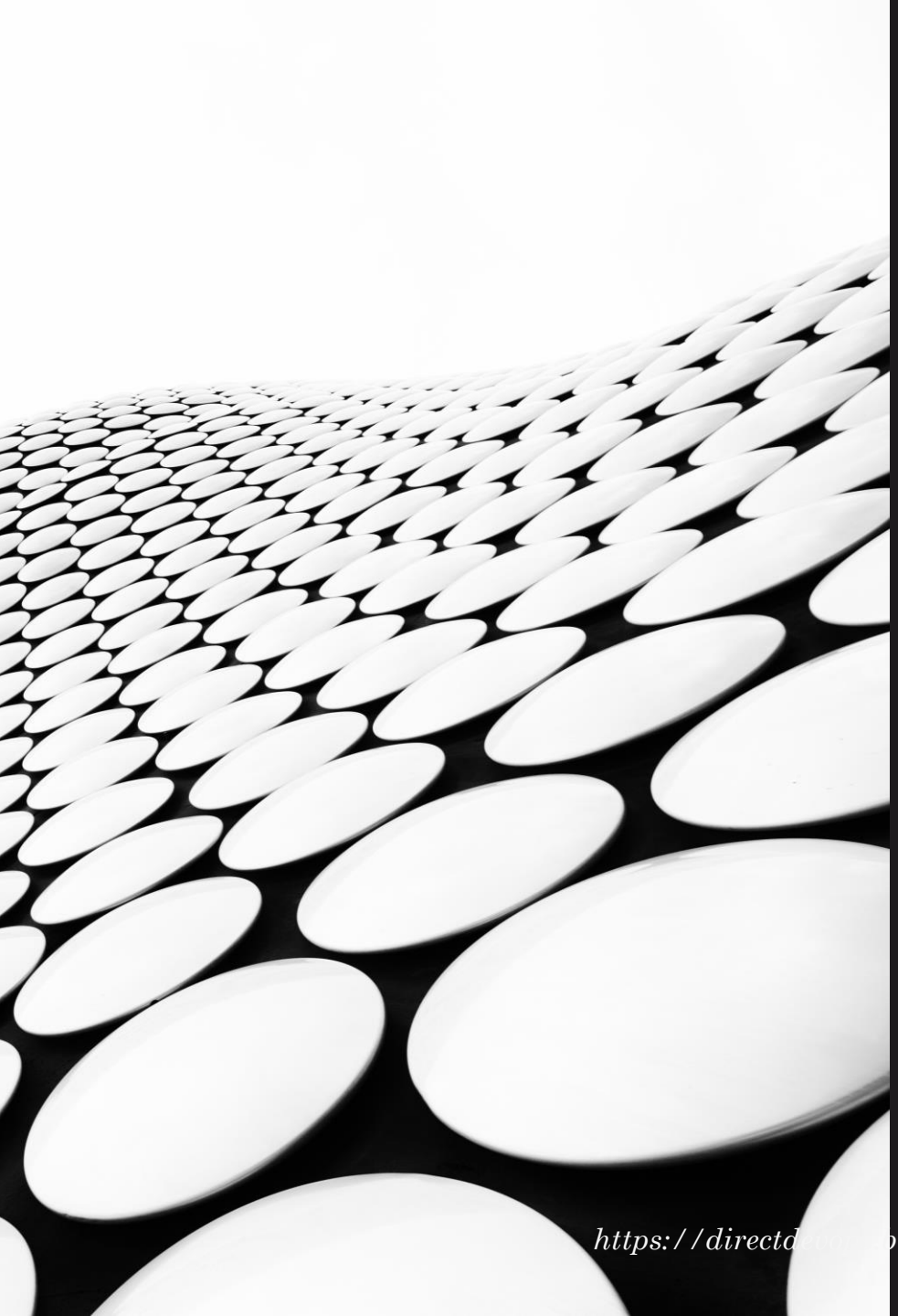


# SITE RELIABILITY ENGINEERING - OVERVIEW

Khaja

# *Site Reliability Engineering Vs DevOps*

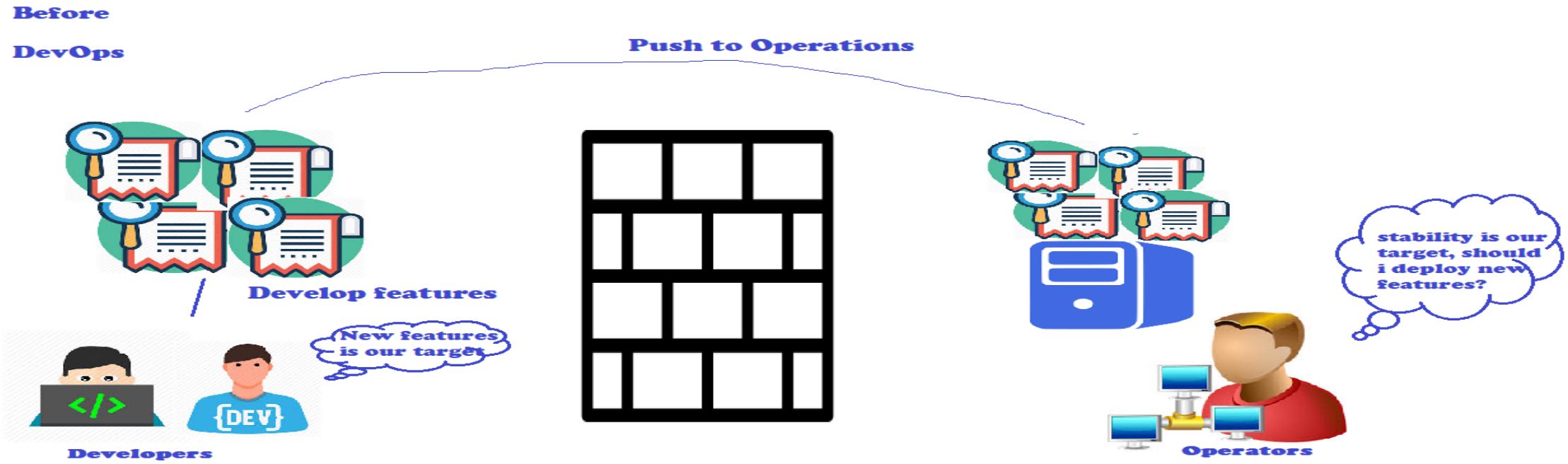
- Are they competing Standards?
- Are they complementing Approaches

# WHAT IS DEVOPS

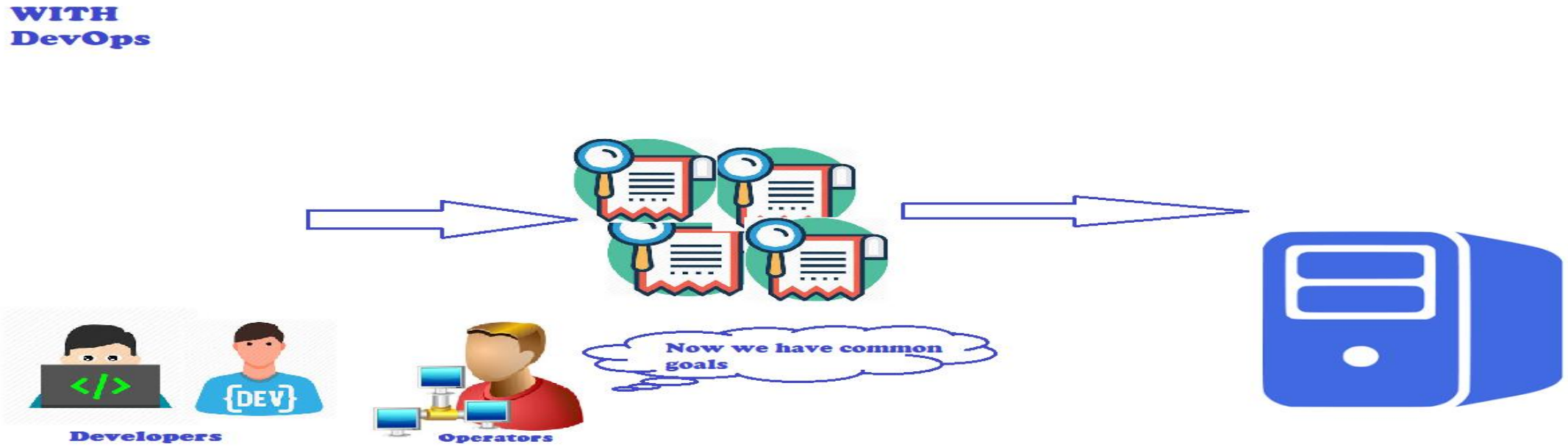
*Before We Answer Who is Better Lets try to understand DevOps*

# *DEVOPS PRINCIPLES*

# *Reduce Organization Silos*




# *SILOS BEFORE DEVOPS*




*DEVOPS BREAKS DOWN THE WALL B/W DEV AND OPS. ALL THE MEMBERS HAVE A COMMON GOAL*

# *Accept Failure as Normal*

- With DevOps we understand Failures are normal, so we would also plan for applying the change & Revert the change

# *Implement Gradual Changes*

- Easier to Deploy with smaller changes
- Finding problems in the case of failures is simple

# *Leverage Tooling and Automation*

- Automate the time taking tasks which help to rollout quickly rather than manually doing the tasks

# *Measure Everything*

- Measure Everything important
- This includes having tangible metrics for evaluation for process & Application level Metrics


# *DEVOPS GIVES THE ABSTRACT IDEA*


# SRE IS ALL ABOUT PRACTICALLY IMPLEMENTATION (CONCRETE IDEA)

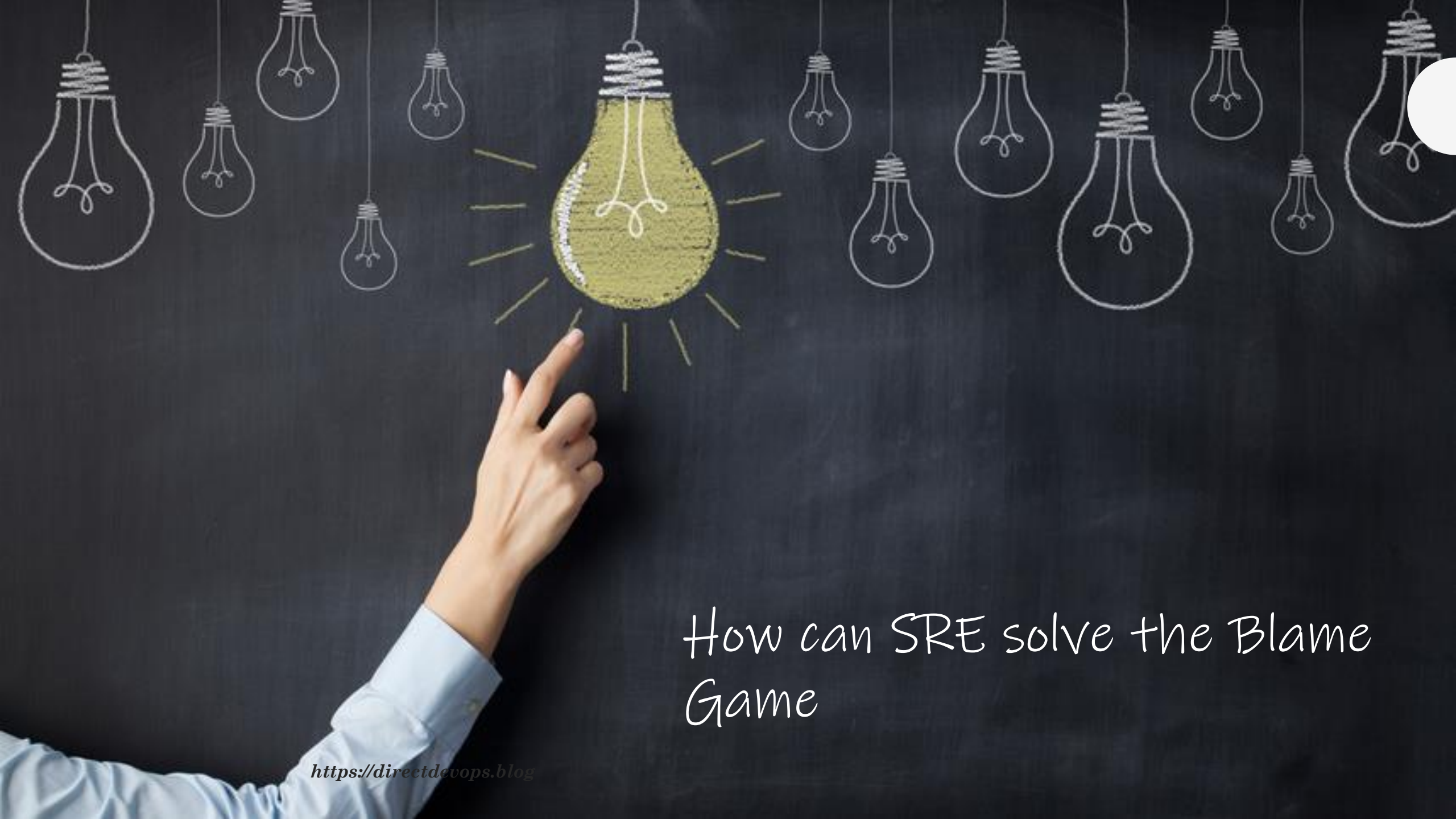
# CLASS SRE IMPLEMENTES DEVOPS




*DEVOPS AND SRE ARE NOT COMPETING METHODS, RATHER THEY ARE CLOSE FRIENDS WITH COMMON GOALS*

# *Who is responsible for failure, Why is System unreliable?*

- Classic Blame between Developers and Operations


How can SRE solve the Blame Game
https://directdevops.blog

# *SRE Discipline*

*SRE's basic principle is 100% is not a real target*

- Define Availability
- Level of availability
- Plan in case of failure

# *Define What Availability is in the context of your application*

- To achieve this We define Service Level Indicator

# *SLI Service Level Indicator*

- Could be
  - Request Latency
  - Batch Throughput
  - Failures per request
- Service Level indicator depicts the metric on the current period of time
- Using SLI Define SLO
- Example:
  - 95 percent of the homepage requests over past 5 minutes have a latency less than 200 ms

# *SLO Service Level Objective*

- Binding target for collection of SLIs
- From SLO we derive SLA
- Example:
  - 99.9% of the requests over a year will have homepage SLI (latency < 300ms)

# *SLA Service Level Agreement*

- More of a commercial agreement, if your service/application is out of spec according to contract
- Example:
  - Customer will receive Service credit if SLI (latency < 300) succeeds less than 99.5% of request over the year.

# *SLIS DRIVE SLOS WHICH INFORM SLAS*




# *RISKS AND ERROR BUDGETS*

# *Risk and availability*

- 100 % availability is not a good idea.
- In many cases user will not experience your availability due to other issues.
- Example:
  - Your service is available for 99.99 % and your customer uses this on a mobile where network availability is only 99%. In such cases due to other reasons user's availability might be equal to less that 99%
- So we have to accept some degree of failures to deliver the features/products
- How?
- That's what Error Budgets are for

# *Error Budgets*

- Acceptable Risk of the system dictates SLO and SLO defines the error Budget
- SLO = 99.9 % available per month
- If you calculate what is time which your services unavailability would be, that defines Error Budgets
- In the above case = 0.1% per month = (0.1* 30 * 24 * 60/100) = **43.2 Minutes/month**
- If the Error Budget is exceeded the feature deployments will be halted, so now it's a common decision which the Product and SRE team has to make when they are deploying risky features
- Error Budgets needs to monitored by a monitoring system


# ARE MANUAL TASKS LIKE RESTARTING SERVICES IN SOME CASES INCLUDED IN ERROR BUDGETS?


# TOIL AND TOIL BUDGETS

# *What is not Toil, rather it is Overhead*

- Emails
- Expense Reports
- Meetings
- I made it explicit because for some of us Toil means the work which we don't like, but that's not the case in SRE

# *What is Toil*

- Toil is kind of work that tends to run production systems that has following characteristic
  - Manual
  - Repetitive
  - Automatable
  - Tactical
  - lacks long-term Value
  - Toils scale linearly as service grows

# *Toil*

- Is Worth Automating
- Scenario:
  - Question: I got an alert in the late night at 2 am and I had to restart the service which took approximately 2 hours. Later during the day from 10am to 4 am I had written a script (an automation around it ) which can avoid the same failure from happening. Are the 2+6 hours considered as toil?
  - Answer: Time spend during middle of night is toil not the time spend to reduce toil further

# *How do we measure Toil?*

- Don’t mix toil and Project work
- Account on call time as toil
- Survey, sample and log toil

# *So what's the goal in terms of Toil*

- Reduce Toil and work on engineering activities
- Eliminating Toils cannot happen or not worth eliminating completing.

# ACTIONABLE ALERTING FOR SITE RELIABILITY ENGINEERS

*https://directdevops.blog*


ALERTS HAVE CAUSED A SLEEP LESS NIGHTS.
https://directdevops.blog


# *THE ALERT FROM MY MONITORING SYSTEM WAS NOT IMPACTING CUSTOMER, BUT IT CONSIDERS THIS FAILURE AS CRITICAL*

*How to solve this using SRE?*

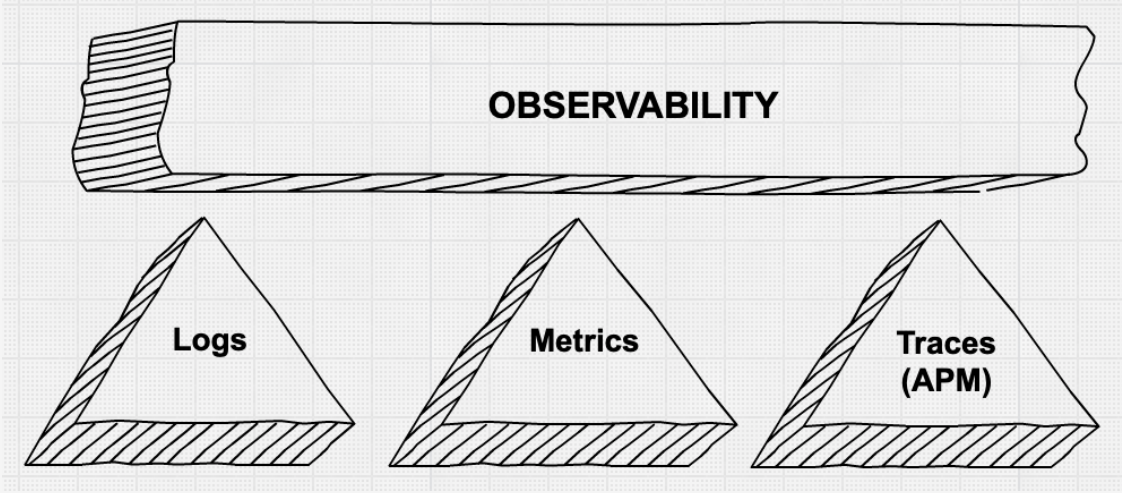
# *How to reduce noise from alerts?*

- Having too many alerts is not good.
- Alert which cause SLIs or SLOs to fail and disable alerts for others (consider that as noise).
- We need fast burn and slow burn alerts impacting your error budgets

# *HOW DO I LET GO THE ALERTS?*

# *OBSERVABILITY*

# *How to achieve Observability*




- We need to make our systems observable.
- Observability breaks down into three key areas
  - Structured Logs
  - Metrics
  - Traces
- Build Dashboards with common starting areas which allows you to drill down further.

# *INCIDENTS*

***https://directdevops.blog***

# *Incident Management*

## SRE suggests the following practices

- Process for declaring Incidents
- Dashboard for viewing current incidents
- Database of who to contact for each kind of incident

## Roles in Incident Management

- **Incident Commander:**
  - Responsible for making key decisions and Plan
  - Assigning roles to others (delegation)
- **Operations Lead:**
  - Who has detailed understanding of system
- **Communications Lead:**
  - Responsible for communicating to external stake holders
- **Planning Lead:**
  - Responsible for Writing the Plan
  - Writing Postmortem document.


# *LEARNING FROM INCIDENTS - POSTMORTEM*

# *Blameless Postmortem Metadata*

- What systems were affected?
- Who was involved in responding?
- How did we find out about the event?
- When did we start responding?
- What mitigations did we deploy?
- When did the incident conclude?

# *Who is involved in Writing Postmortem document*

- Initially Incident Commander is involved in writing a draft of this document
- And later the document is encouraged to be collaborated by all the people involved in incident.
- Remember System Failure should be treated as failure in System rather than Blaming humans
- Write everything that contribute to failure rather than one root cause.