Opsgenie

Scaling On-Call in a DevOps Organization

Contents

On-Call Fundamentals

Just like emergency rooms require on-call schedules for doctors to handle health emergencies, DevOps teams need on-call schedules to efficiently respond to software and system issues that impact performance, deployment, and availability. These schedules ensure issues are not missed and are assigned to people with the skills to effectively evaluate and initiate a plan of corrective action.

Generally, a team of engineers is placed on “on-call” and if there are disruptions during a rotation, the team is notified. The best on-call schedules have redundancy with at least primary and secondary responders. If the primary responder is unable to take action in a designated time-frame, the secondary responder can be notified.

Modern on-call and alerting solutions like Opsgenie automate the notification process. Opsgenie uses flexible schedule options which are key for handling issues in a complex environment, and can be adapted from workflow to workflow. When there is an alert, on-call engineers get notified through channels such as SMS, voice call, push notification or email. As a notification is received, the alert is usually marked as acknowledged - meaning that someone got the notification and has started to work on this issue. Eventually, the on-call employees identify the problem and resolve it. As a best practice, after resolution, there is analysis to determine the cause of the issue and opportunities to avoid recurrence and accelerate response.

On-Call Concepts

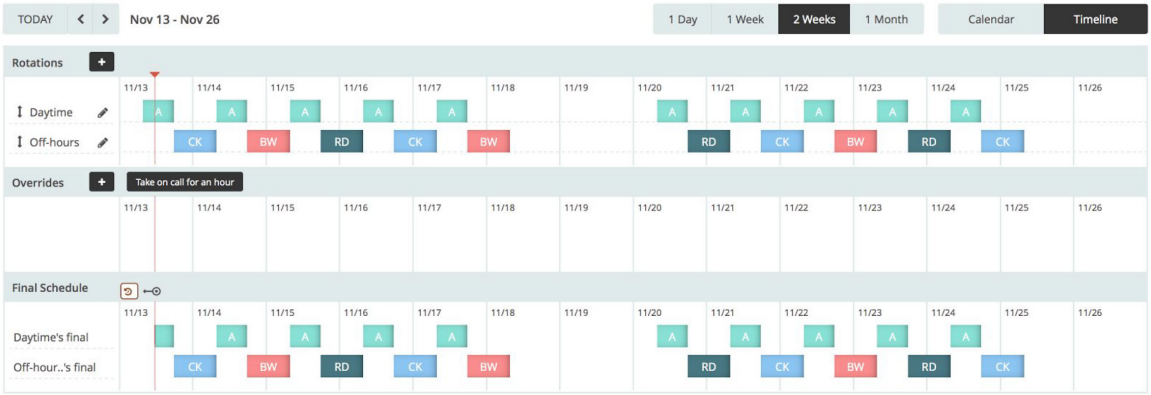
WHAT IS AN ON-CALL SCHEDULE?

On-call schedules are used to determine who is on-call at a given time. On-call scheduling tools need to be flexible enough to support sophisticated scheduling scenarios as there can be very complicated scheduling requirements due to geographically distributed teams, schedule changes, team, and personnel changes.

In Opsgenie, an on-call schedule usually has one or more rotations and an additional rotation to support overrides. A **rotation** represents a set of people who rotate on-call responsibilities through the same shift. **Override** layers allow a user to replace an on-call user for a specified amount of time. **Final schedule** combines the rotations and override layers and shows the final schedule.

In this on-call schedule example, we have five engineers: Bruce, Peter, Charles, Scott and Clark.

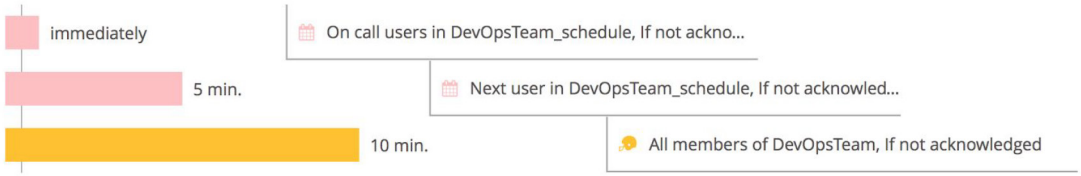
During the week, Bruce, Peter and Charles share on-call duty rotating daily, starting from 9:00 am until 9:00 am the next day. On the weekends, Scott and Clark are on-call, rotating daily from Saturday morning 9:00 am until Monday morning 9:00 am.



In this example, the on-call schedule is based on business hours and off-hours. Here, “A” in the Daytime schedule represents the whole Application team. During business hours (week- days 09:00 - 18:00), all members of Application team are notified for alerts. In the evenings, Clark, Bruce and Raven share the on-call duty, rotating daily.

WHAT IS AN ESCALATION

Escalations determine who should be notified when there is an alert and what should happen if nobody responds to that alert. In other words, escalations are used to define who should be notified next, in which order, and through which timeline.



In this simple example, on-call engineers of the **DevOpsTeam_schedule** get notified immediately when a new alert is created. If this team does not respond to an alert in 5 minutes, second escalation step gets activated meaning that the next person on **DevOsTeam_schedule** schedule gets notified. If 10 minutes has passed since the creation of the alert and still no-one has taken any alert action, then the third escalation step is executed and all members of the **DevOpsTeam** team are notified.

The DevOps Movement and Its Crucial Use of On-Call Scheduling

Up until the last decade, responding to IT incidents was the primary job of the Operations teams. Organizations typically had a tiered team structure (i.e. Level 1, Level 2, Level 3) to respond to issues reported by customers or monitoring tools. The main goal in adopting this structure was to reduce operations cost. Usually, Level 1 would involve entry-level employees which was cost effective. If Level 1 was not able to resolve an issue, the issue would be escalated to Level 2, made up of more senior and therefore, expensive, people. This process would continue until the issue was resolved.

Recently, with the rise of “always on” services, customers expectations have risen. There are now greater interdependencies between systems. Urgency has become more important because a slow response to outages can impact a company’s reputation due to social media. As a result of these factors, the structure of response teams needs to change and addressing incidents in a timely manner is mission critical. The DevOps movement set out to address this goal. Being most familiar with the code, developers are the ones who can best troubleshoot related issues in the shortest amount of time. Thanks to the DevOps movement, it is now common practice for developers to be on-call, ensuring the code runs well, lowering the MTTA and MTTR of alerts.

When the development team shares operational responsibilities, there are added benefits beyond the reduction in MTTR. Code is tested more rigorously, since that the team may be notified off hours if it has issues. In short, there is an ownership shift of mind and more reliable and resilient systems are built as a result.

On-Call Schedule Patterns For Growing Companies

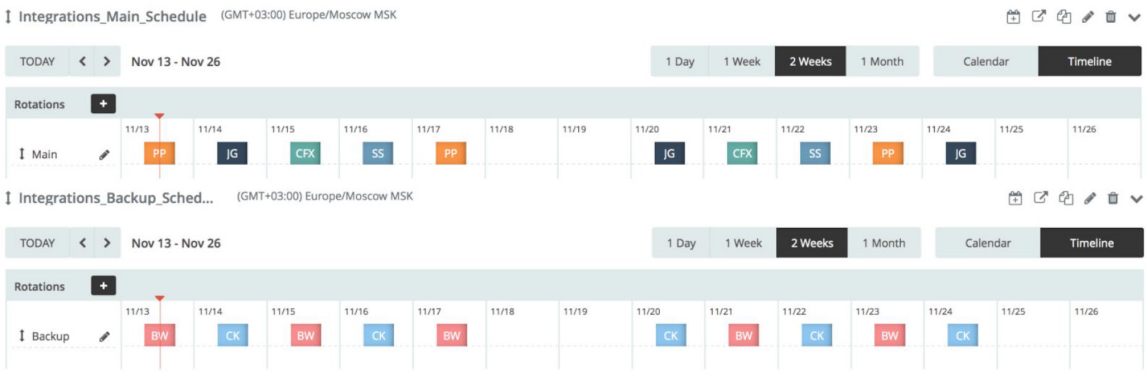
Another reason on-call is becoming a more vital concept and on-call teams have to scale is the growth of organizations. As organizations grow, it is not just the number of employees that grow, but also the products, services, and the operational workload. Thus, it becomes a big challenge to continue to produce new features and grow products as well as make sure that the systems are healthy and up all the time. No surprise, all of these exciting complexities and changes naturally result in a change in organizations' on-call approach as well.

ONE OR MULTIPLE PEOPLE ON-CALL

In case of small organizations, usually, one person is on-call at a time and that person is responsible for the health of the whole application or system. As organizations add new people, expertise teams are usually built where each team is responsible for one or a few domains or microservices of the whole system. In the case of big organizations, there is a tendency to have on-call people from each team at the same time to make sure that there is good coverage in terms of the various skills and domains. Overall, this means having more people involved in on-call duties.

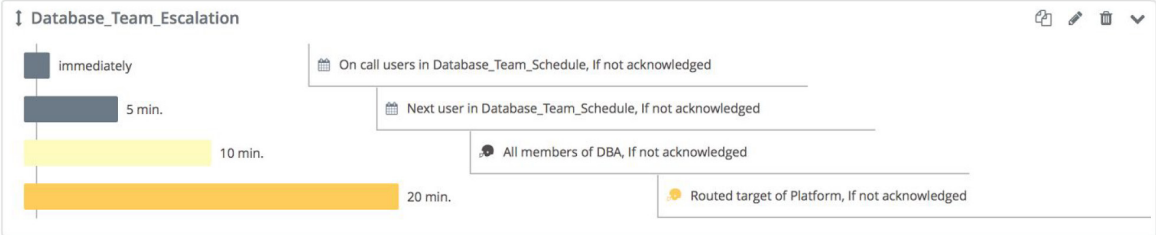
MULTIPLE ESCALATIONS

A very useful approach is to make sure there are multiple escalations available in each on-call schedule, so that the primary responders can get the help of others, as needed. This is especially good in case of growing companies, where there could be people on-call who are not experts of the systems yet. Here is an example on-call schedule with multiple escalations from our own usage of Opsgenie. We make sure that we have a backup schedule and that an experienced engineer is always available for additional support.



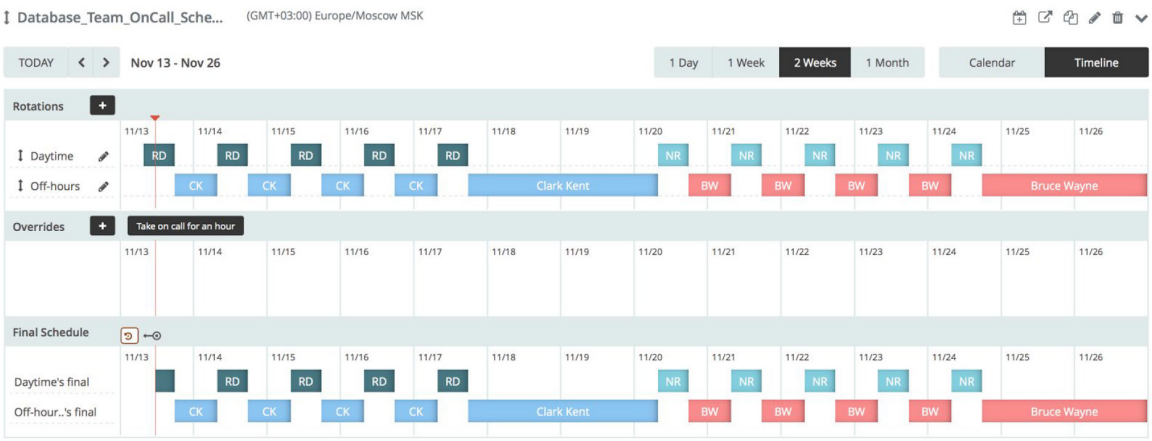
ESCALATING TO A DIFFERENT TEAM

Another pattern we observe from our customers' usage is related to escalations. There is usually a first escalation of unattended issues to the back-up people in the schedule, then to other people in the same team. If the issue is still not taken care or resolved, another team helps. In the example below, unresolved issues of **Database** team get escalated to the **Platform** team.



SEPARATE NIGHTWATCHERS

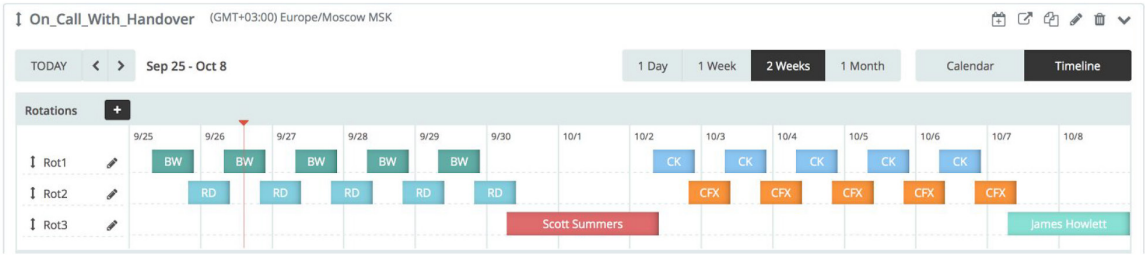
One pattern we see in our customers' usage is to have different set of people on-call during business hours and off-hours.



In the example above, note that the nightwatcher is on-call during the weekends as well.

SCHEDULES WITH OVERLAPS

Some companies make sure there are overlaps between shift changes so that the previous on-call engineers can have some time with the next ones and can delegate the work and communicate properly.



FOLLOW THE SUN SCHEDULES FOR INTERNATIONAL ORGANIZATIONS

Some companies have international teams and can have 24/7 coverage easily within business hours, as teams have different time zones. In the example below, there are on-call engineers both located in Europe, in the U.S. and in Australia. An engineer from each location covers the on-call duty during business hours, each for 8 hours a day.

In the example below, there are on-call engineers in India and in the U.S. Each engineer has on-call duty for 12 hours a day.

Advice to Management Teams While Enrolling Changes to On-Call Systems

Being on-call can be a daunting and disruptive experience. Many people with on-call duties complain how having to be ready to handle incidents affects work life balance, even health, as on-call employees may be frequently woken up in the middle of night or may need to plan evenings and weekends while considering on-call duties. As organizations enroll changes to scale on-call teams, it needs to be considered how to best match that evolution with a sustainable and humane solution. Below is some advice based on our experiences so far with our customers.

1 TRANSPARENCY

Transparency is the key to successful communications. Clarifying expectations around availability of employees is a must when rolling out an on-call system or a change to an existing on-call system. People with new on-call duties may have questions like the following; the answers should be clear before enrolling the changes:

- Are engineers supposed to be on-call during nights?
- If on-call during nights, is there flexibility to work from home the next day?
- Or start the next day later than usual to make up needed sleep from the on-call part of the night?
- Are engineers supposed to do development work during on-call time?
- Maximum how many times in a month would an engineer be on-call?

2 TRAINING

Providing proper training to new on-call engineers is a must for scaling on-call systems successfully. Having an established training plan for on-call processes and tooling is the first step in this direction.

Providing engineers with detailed and up-to-date runbooks is also very important. That helps engineers see what the protocol is for certain situations, or how similar issues were resolved effectively in the past.

Shadowing experienced on-call engineers would be a valuable experience as well, so that the new on-call engineers can feel the on-call atmosphere directly while still being given some guidance and support.

Having multiple escalation channels on on-call schedules is a good idea especially for junior on-call engineers. Not being completely alone when help is needed will come as a relief. Related to this, having the junior engineers in the primary on-call rotation and having a more senior engineer as back-up or in a secondary rotation is another best practice. This organization helps junior engineers develop the required on-call skills while avoiding panic when there is an issue beyond junior expertise.

3 DEVELOPMENT DUTIES DURING ON-CALL

Having development duties during on-call usually means lots of context switching and interruptions, especially when the on-call duty is a heavy one. It also has negative effect on development sprints since it is difficult to estimate how much contribution on-call people may have to the development sprint. This usually means less development efficiency and more stress on the on-call engineers. As a best practice, we recommend not assigning development duties to developers during on-call duties and when there is free time, request for the developer to work on improving the on-call related documentation and automation to eventually improve the sustainability of the systems and services.

4 WELL-DEFINED AND FINE-TUNED PROCESSES AND SYSTEMS

A healthy on-call system can only exist if improved constantly by fine-tuning processes and systems. A few recommendations towards this goal:

- Evaluate alert priority and urgency and set systems based on that. Low urgency alerts can wait until the morning; on-call employees do not need to wake up in the middle of the night because of these alerts.
- Reduce false-positives by classifying alerts based on factors such as root-cause, originat ing system, message, thresholds etc. This way, differentiate actionable alerts from the rest.

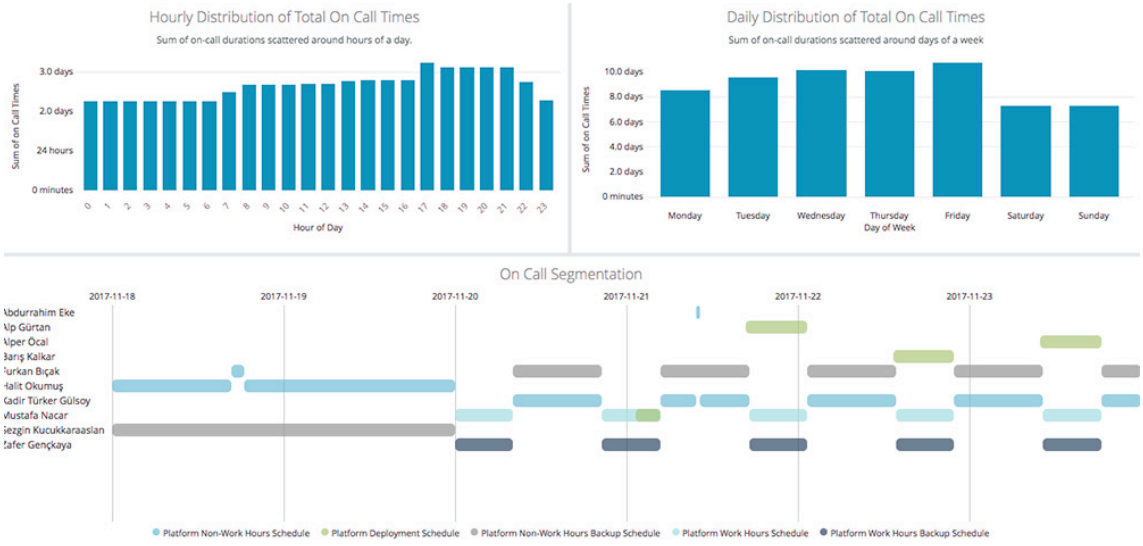
- Deduplicate related alerts to avoid alert fatigue; on-call employees do not need to get notified by a flood of alerts in an hour for the same issue; give the chance to focus on the resolution instead of notifying with the same alert over and over.

- Design rich alerts that empower the on-call engineers to make effective decisions and apply the knowledge recorded in runbooks.
- Provide alert reports and metrics to on-call teams so that the weak areas can be seen in the systems and insights gained on the root-causes and resolutions. Do not let on-call teams get bogged down by the same problem repeatedly.

5 REVIEWING ON-CALL REPORTS

For fairness, managers are advised to review on-call related reporting and check how often each team member was paged or woken up, and enroll changes in the on-call system considering such facts and hopefully avoid employee burn-out. Below is an on-call report for one of our teams internally. The report shows information such as how long each team member has had on-call duty, distribution of on-call hours for each on-call schedule, and the hourly and daily distributions of on-call duty for each person.


User On Call Times
Run
FILTERS
Time Period is in the past 7 days
User is any value
Schedule is any value
Rotation is any value
46.0 days
Total On Call Time
34.7 days
Total (Non-Overlapping) On Call Time
On Call Times
10.0 days
8.0 days
6.0 days
4.0 days
2.0 days
0 minutes
Halit Okumuş
Abdurrahim Eke
Alper Öcal
Sezgin Kucukkaraaslan
Furkan Bıçak
Mustafa Nacar
Kadir Türker Gülsoy
Zafer Gençkaya
Alp Gürtan
Barış Kalkar
User name
Non-Overlapping On Call Time of User
Overlapped On Call Time of User
On Call Times per Schedule
Platform Non-Work Hours Schedule
53%
Platform Deployment Schedule
29%
29%
29%
12%
Platform Non-Work Hours Backup Schedule
46%
54%
Platform Work Hours Backup Schedule
100%
Platform Work Hours Schedule
100%
Abdurrahim Eke
Alp Gürtan
Alper Öcal
Barış Kalkar
Furkan Bıçak
Halit Okumuş
Kadir Türker Gülsoy
Mustafa Nacar
Sezgin Kucukkaraaslan
Zafer Gençkaya


Hourly Distribution of Total On Call Times
Sum of on-call durations scattered around hours of a day.
Sum of on Call Times
3.0 days
2.0 days
24 hours
0 minutes
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day
Daily Distribution of Total On Call Times
Sum of on-call durations scattered around days of a week
Sum of on Call Times
10.0 days
8.0 days
6.0 days
4.0 days
2.0 days
0 minutes
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of Week
On Call Segmentation
2017-11-18
2017-11-19
2017-11-20
2017-11-21
2017-11-22
2017-11-23
Platform Non-Work Hours Schedule
Platform Deployment Schedule
Platform Non-Work Hours Backup Schedule
Platform Work Hours Schedule
Platform Work Hours Backup Schedule

6 FRIENDLY ON-CALL CULTURE

On-call engineers carry huge responsibility for the success of companies. The job requires extra caution when rolling out changes, and responding knowledgeably as fast as possible when there is a problem. Such responsibility may naturally mean stress and tension, especially when there is a big issue and there are unknowns. The on-call culture set by the senior on-call engineers and management teams defines how people deal with that stress and tension and whether the on-call experience is transformed for the good ofthe company or not.

Lastly, we recommend management teams make sure that on-call employees are being listened to. Management should organize all-hands meetings with the on-call engineers to discuss problems, complaints, and areas of weaknesses. Make sure to take actions which help resolve these problems and weaknesses. Constantly reevaluate the on-call system, tools, processes, people, documentation, and training plans that are being used. Lastly, make sure that management is not the sole decision maker when it comes to on-call organization and protocol, and instead, make each decision a team decision.

For both the on-call engineers' sake and the on-call culture of the company, management teams should pay attention to developing a friendly on-call culture and make it clear that the goal should always be to find the problems, risks, and weaknesses in systems and solve them. The goal should never be to find whom to blame. Opsgenie offers the tools to make on-call schedules, processes, and experiences easier and provides all the opportunities to continue improving at every turn.


Opsgenie