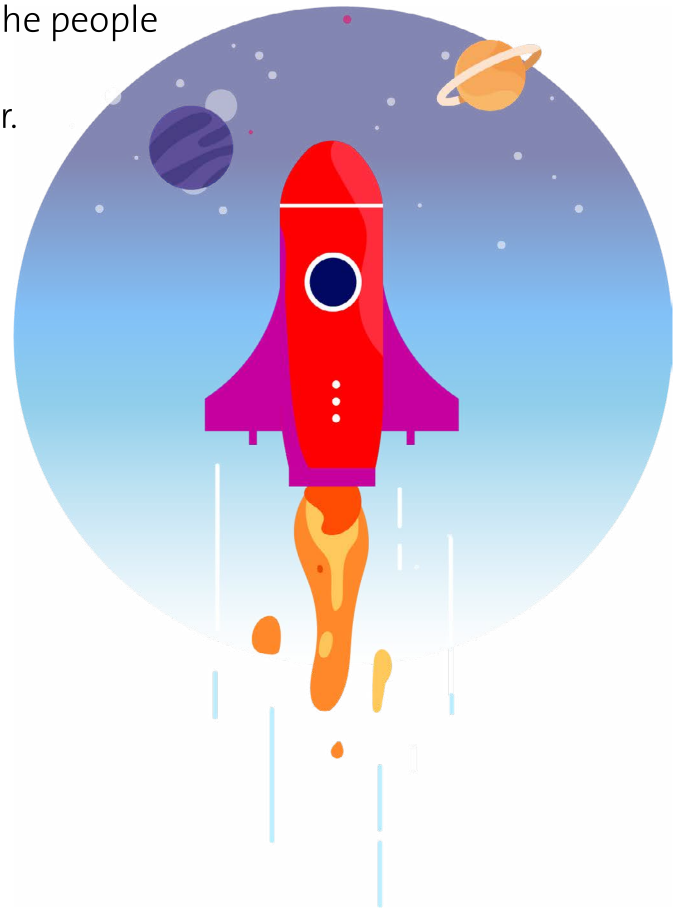
DevOps Maturity Check

swisscom

DevOps Maturity Check

Introduction

The DevOps Maturity Check is a tool that helps teams identify their development potentials and increase their DevOps maturity. It is completed by the entire team. By team we mean the people and their roles. Whether it is a virtual or organizational team does not matter.



What is the DevOps Maturity Check?

The DevOps Maturity Check is based on a self-assessment of the team. The purpose is to show areas of improvement in the process and with that to give the team the possibility to correct and improve. For example if the DevOps Maturity Check shows that the test automation is too low the team can set a specific and measurable goal to improve that (e.g. xy % of all unit-test cases can be run fully automated). The DevOps Maturity Check does not deliver objectively measurable and comparable results. It is not made for comparing teams or to set quantitative goals.

Who is the DevOps Maturity Check for?

The model is designed for all teams with an agile or DevOps working model. This is independent from their organisation affiliation, size, or their maturity goals.
Practice has shown that the DevOps Maturity Check helped not only experienced developer teams but also non-technical consulting teams.

DevOps Understanding at Swisscom

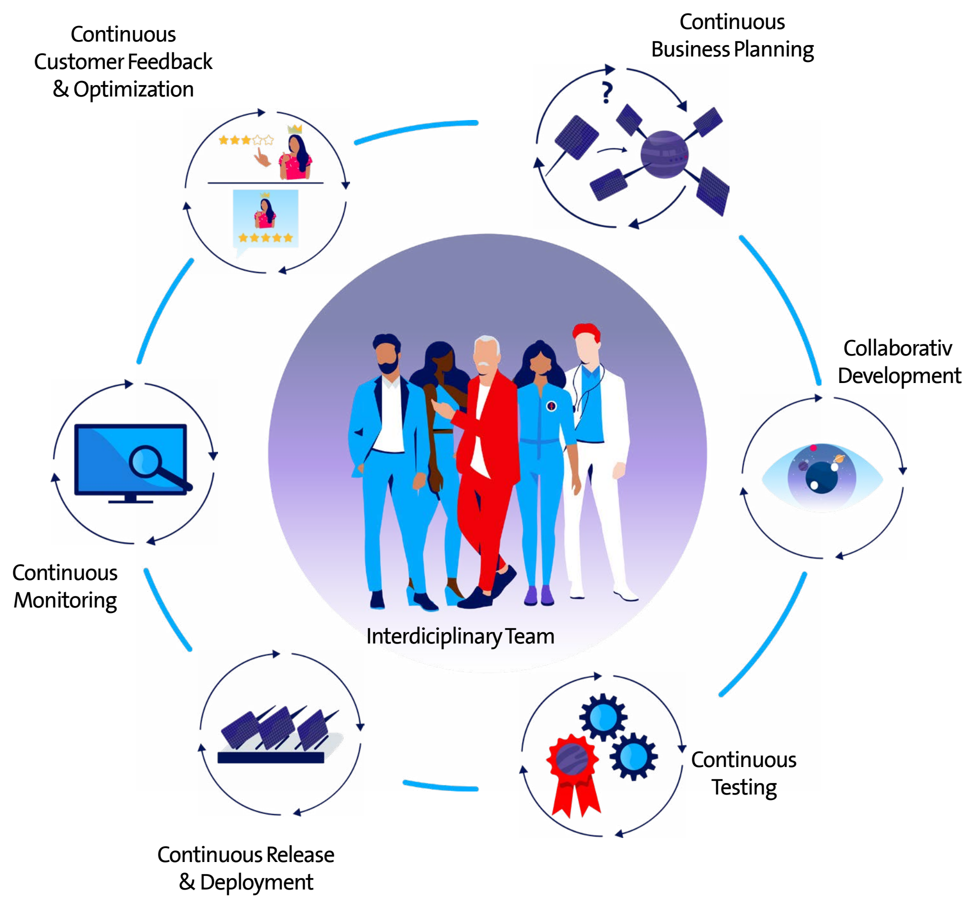
DevOps at Swisscom

The maturity check reflects our understanding of DevOps and agility. Swisscom has chosen a holistic approach from the very beginning. In addition to the obvious technical capabilities, the organizational form, processes and above all the culture have a significant influence on the success of DevOps.

Technical Capabilities - The Reference Model

The IBM reference model „DevOps: The IBM approach - Continuous delivery of software-driven innovation“ provides a good overview of the required technical skills.[1]
Swisscom relies on an adapted version of the IBM reference model. The company uses it as a holistic system that covers not only development, but all relevant skills along the entire value stream. The Swisscom model describes skills that are relevant for a DevOps operation, however, it is not a flow chart.

Collaborative Development:
This term describes the ability to continuously generate value in short iterations and interdisciplinary teams consisting of representatives from business, development, security, testing and operations. This requires that code is continuously fed into the code repository - in the best case several times a day. Everything is considered code - this includes not only the actual source code, but also infrastructure, deployment scripts, monitoring, test data and more.


Continuous Customer Feedback & Optimization
Continuous Business Planning
Collaborativ Development
Continuous Monitoring
Interdisciplinary Team
Continuous Testing
Continuous Release & Deployment

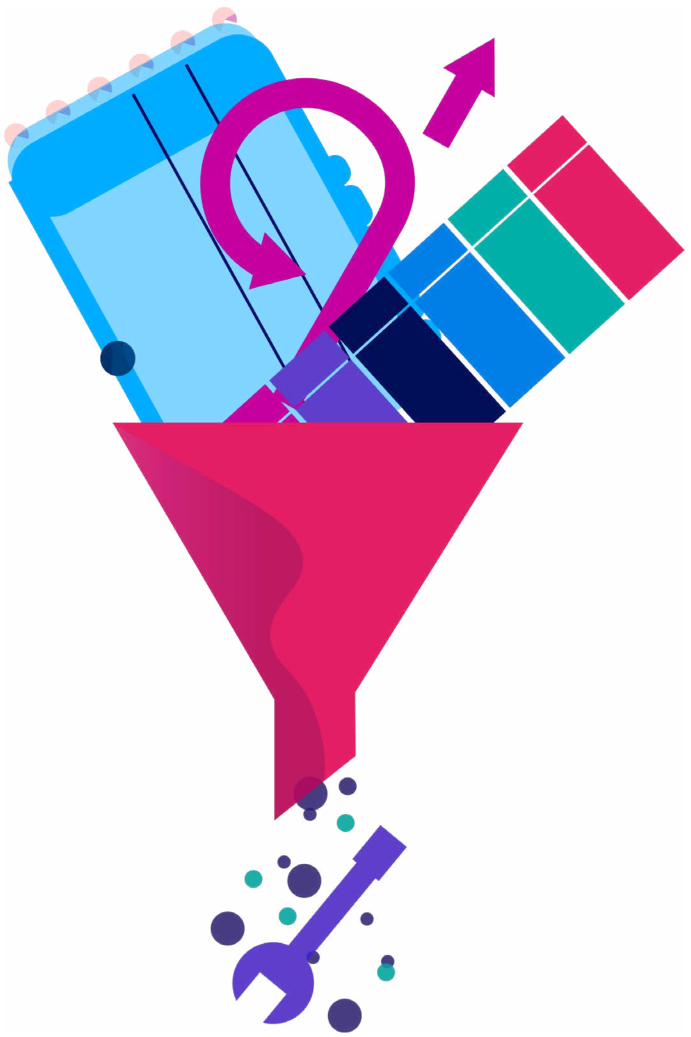
Continuous Testing:
This includes the ability to test the code in a reproducible manner and repeatedly. Automatic tests allow to examine it in many ways immediately after checking it into the repository. This allows early and continuous testing instead of at the end of a process chain, shortly before the introduction into production. This makes testing part of the overall process. This in turn improves cost efficiency, since errors can be eliminated early in the cycle and thus at a lower cost.

Continuous Release & Deployment:
This refers to the ability to automatically execute a build and deliver it to any environment. This is necessary to realize the full potential of automated testing. Code check-in in the repository allows a number of automated processes to be performed, from build to test to deployment. This significantly reduces manual activities.

Continuous Monitoring:
This involves monitoring the entire system on all platforms at all times. This makes it possible to see immediately after the installation of a feature whether it affects the performance of the entire system. This early feedback helps to further improve quality.
Continuous Customer Feedback and Optimization: This refers to the ability to continuously improve software based on measurements and feedback from both the customer and the company. Customer behavior is deliberately analyzed and various channels are used to obtain early and comprehensive feedback.

Continuous Business Planning:
This refers to the ability to react quickly to customer needs and to continuously incorporate appropriate feedback. The focus is always on customer needs and defines the product.

[1] DevOps: The IBM approach - Continuous delivery of software-driven innovation (especially page 6), URL: developer.ibm.com/community



Organization

Swisscom relies on interdisciplinary, fully dedicated, autonomous teams that communicate with each other in a self-organized manner and at eye level. The ideal team size has proven to be 7 - 10 people who work together as stably as possible over a long period of time, thus developing a common learning culture and being as self-organized as possible. Project teams that were formed at the beginning of the project and dissolved again at the end have been replaced by standing product teams. In this way, the person doesn‘t goes to work, but the work comes into the team.

Processes

A framework like Scrum, Kanban or the Scaled Agile Framework © is always just a means to achieve the goals. These frameworks must never be an end in themselves. That is why we have consciously decided in the Maturity Check not to check the framework‘s maturity, but rather the underlying values and principles that we consider to be much more important

Culture

Working in an agile environment requires different behaviors on different levels. At the individual level; more personal responsibility, initiative and open feedback behavior are desired. The team should take joint responsibility, a common learning culture should be promoted and high quality services should be provided. And at the leadership level, the team should be granted openness, trust and the greatest possible autonomy. We have summarized the corresponding expectations in a Collaboration Charter, which is based on the corporate values.

Individual

I continue do develop as my own brand and am open to new tasks. I have professional pride. I take care of my own well-being, keep my promises, and take responsibility for my results. I give constructive criticism. I strive for continuous improvement and support my team and the company. I am receptive to change and support my team and the company in continuing to respond to customer needs. I deliver cost-efficient, reusable, secure and straightforward solutions. I do the right thing and take action if I think my leaders or colleagues are in the wrong.

Team

We create a framework that allows us to work quickly and independently, and support interdisciplinary collaboration. Together, we keep our promises and take responsibility for our team results. We are proud of our built-in quality and celebrate successes. We practice a no blame culture, which encourages open and honest feedback. We treat one another with respect and support everyone’s right to a healthy work-life balance. Mistakes are part of our shared learning process. We resolve problems for the long term. We abide by the team and company rules and regulations. We take full responsibility over the entire life cycle of our products and services.

Lean Agile Leadership

At Swisscom, everyone leads. As a lean agile leader, I demonstrate entrepreneurial spirit and am a trainer of the organisations. I create an agile environment in which people are able to do great work, and facilitate autonomy while also giving clear direction. I keep our promises and take responsibility for our team results. I believe in the team and trust that it has the skills to develop and deliver great products. I keep rules and regulations to a minimum and demand compliance. I provide open, timely information about current processes and therefore lead by transparency instead of control.

DevOps Maturity Check Principles

Self Assessment

The team should judge itself.
The statements in the Maturity Check help the team to convey our DevOps understanding. They are discussed, reflected and assessed together in the team.

Of course, self-evaluations are always subjective and consequently never comparable. Swisscom's Maturity Check does not serve to compare teams and is not used for this purpose.

It's about action

An important element of the Maturity Check are the measures: What actions do we as a team derive from the check and by when do we want to achieve them?
A maturity check without consequences is a waste of time.

Collaboration

The DevOps Maturity Check can be used to collaborate with other teams. It is meant to encourage teams to reach out to each other, exchanging experiences and practices. This way, it provides an opportunity to learn from and grow with each other.

Two Dimensions

The Maturity Check consists of two independent parts: Team Fitness and Application Fitness. The Team Fitness deals with its collaboration, working methods and the use of agile principles. The Application Fitness checks how well the DevOps Capability is implemented.

How do I use the DevOps Maturity Check?

The DevOps Maturity Check provides a set of statements that the team either agrees with or doesn't. They are assessed and discussed in the team. A retrospective is a good space to do this.

To help assess where you stand there are some examples of what needs to be fulfilled in order to achieve a rating of 5.

The assessment ranges from 0 to 5, 0 being "not at all" and 5 being "very accurate". The PDF is interactive, so you can rate the question here directly.

We live the principles in the „Team" part of the Collaboration Charter. (see chapter „DevOps Understanding at Swisscom")

- ○ We regularly address the subject of charter as a team. We deal with the content.
- ○ We continuously improve ourselves.
- ○ We consistently live the behavior patterns described and examples of this are visible in everyday life:
- ○ „We create the prerequisites for working speedly and independently and foster interdisciplinary collaboration."
- ○ „Together, we deliver on our promises and assume joint ownership for our team results."
- ○ „We are proud of our built-in quality and celebrate success."
- ○ „We live a "no blaming" culture that encourages open feedback."
- ○ „We treat each other with respect and we support everyone's work-life balance."
- ○ „Mistakes are part of our collaborative learning process."
- ○ „We remedy problems sustainably."
- ○ „We adhere to the rules of the team and company."
- ○ „We take full lifecycle responsibility for our products and services"

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Team Fitness

The first part of the DevOps Maturity Check deals with collaboration, working methods and the application of agile principles in a team.

Working Together
Self-organized teams live a culture that promotes psychological safety. They have the desire to constantly improve themselves and thus deliver the best results while having fun at the same time.

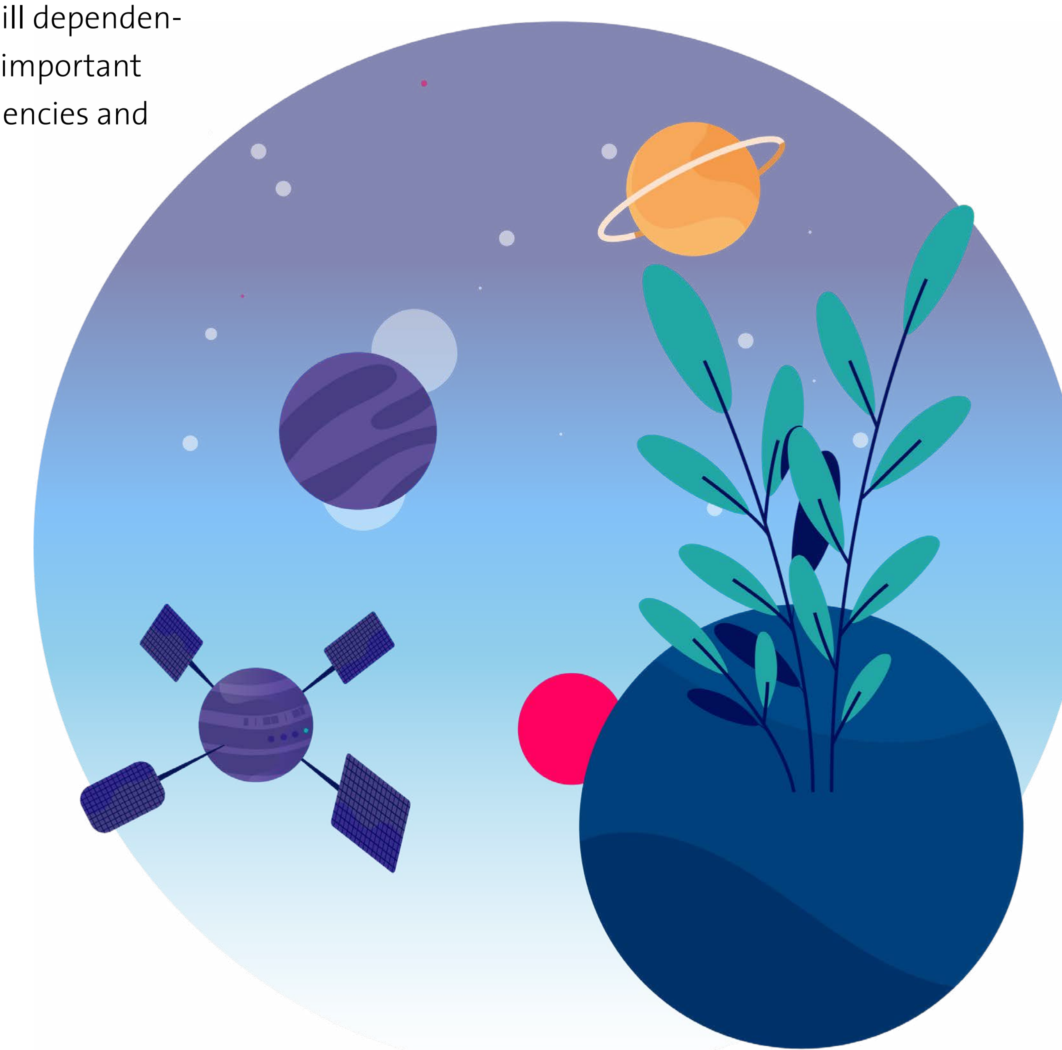
Value-Oriented
Teams focus on an important agile principle - the value orientation. Aligned with a clearly communicated product vision, hypotheses are quickly and effectively validated to achieve the defined business outcomes.

Team Setup
Agile teams work together consistently over a long period of time. Within the team, the skills needed to deliver the service are available and one role (e.g. the Scrum Master) takes care of the processes and practices defined in the team.

The team takes the end-to-end responsibility for secure solutions from development to operation.

Way of working
Agile is not chaotic or anarchic - concrete results are delivered in short intervals.The team follows a prioritized backlog that can be viewed by all stakeholders. based on experience, teams can ensure reliable forecasts.

Ecosystem
Especially in very complex systems and with older architectures, there are still dependencies to other departments. It is important that teams know these dependencies and manage them well.



Collaboration

We live the principles in the „Team“ part of the Collaboration Charter.
(see chapter „DevOps Understanding at Swisscom“)

○ We regularly address the subject of charter as a team. We deal with the content.
○ We continuously improve ourselves.
○ We consistently live the behavior patterns described and examples of this are visible in everyday life:
○ We create the prerequisites for working speedly and independently and foster interdisciplinary collaboration.
○ Together, we deliver on our promises and assume joint ownership for our team results.
○ We are proud of our built-in quality and celebrate success.
○ We live a “no blaming” culture that encourages open feedback.
○ We treat each other with respect and we support everyone‘s work-life balance.
○ Mistakes are part of our collaborative learning process.
○ We remedy problems sustainably.
○ We adhere to the rules of the team and company.
○ We take full lifecycle responsibility for our products and service.

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Thanks to continuous reflection in the team, we constantly and measurably improve performance, cooperation, and working methods.

○ After each iteration we take time to reflect on our performance, cooperation, development and way of working.
○ We document results within the team and make them transparently accessible.
○ We define improvement measures, document them and implement them.
○ We make the expected performance improvement transparent: in addition to the actual measure, the expected result (hypothesis) is also defined.
○ We measure the expected improvement in performance and thus demonstrate its impact.

Rating

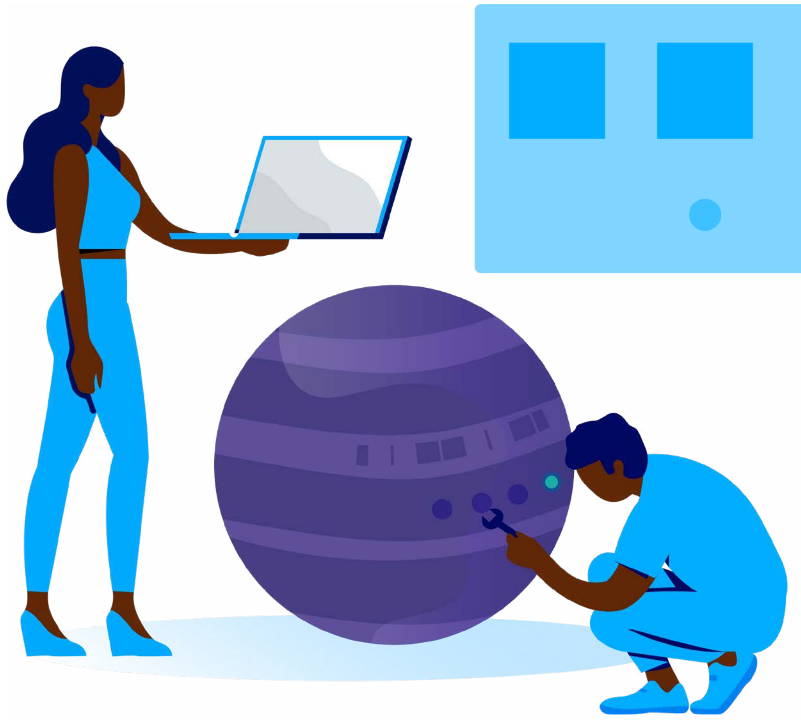
○	○	○	○	○	○
0	1	2	3	4	5

We work self-organized. We bear responsibility together.

○ We can give examples of how and where we have shared responsibility.
○ As a team, we decisively determine HOW something is implemented and can influence the „WHAT“.
○ As a team, we decide how much work we want to do and how this is coordinated
○ We have defined our decision-making process and it is applied by everyone as needed.
○ All decisions can be made by everyone in the team.

Rating

○	○	○	○	○	○
0	1	2	3	4	5



Value Orientation

We have a business vision that is central and from which we derive clearly defined business goals.

○ The business vision exists and is known to all.

○ This enables us to generate value for our company (shorter time to market, greater flexibility in product design, higher quality, lower production and operating costs, etc.).

○ We have derived quantifiable Key Performance Indicators from this, measure them regularly and make the results available to everyone in a transparent manner.

○ The results of the KPIs are incorporated into the iteration planning.

○ The KPIs are balanced and consider all relevant dimensions (business value, security, stability, costs, ...).

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Our work is based on hypotheses (falsifiable assumptions) which we validate as effectively as possible.

○ Based on our observations and experiences, we make hypotheses.

○ We have clearly defined these hypotheses and made them visible to all.

○ We test these hypotheses by carrying out simple, cheap and quick experiments.

○ The result of the experiment flows into the further development.

○ Behind every development and every project there is a hypothesis.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

We have a client or product manager/owner who continuously prioritises customer requirements on behalf of one or more business units and aligns them to business value.

○ The product manager/product owner/client is operationally involved, i.e. they participate in the respective team rituals.

○ They are promptly available to the team for questions.

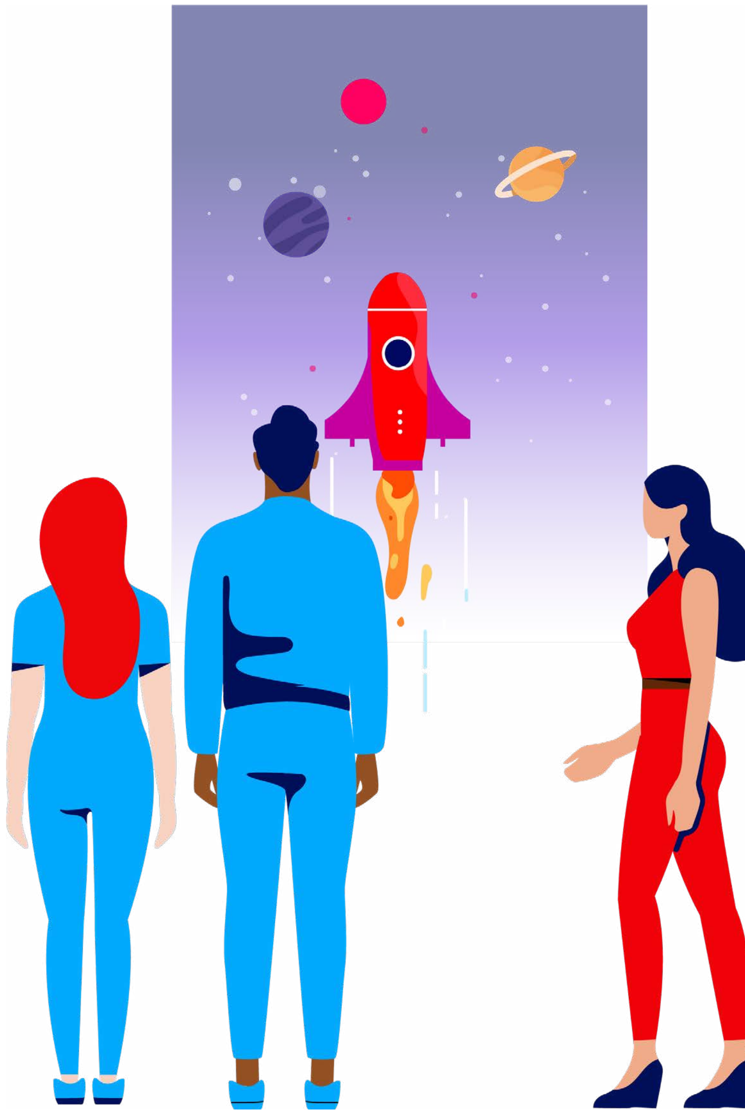
○ The product manager/product owner/client regularly prioritizes the work in order to generate the greatest possible business value.

○ They can make necessary business decisions at any time and in consultation with all stakeholders.

○ The customer/product owner is a person and not a group or impersonal organizational unit.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5



Team Setup

We have one person in the team, such as the Scrum Master, who takes responsibility of ensuring that the values, processes and practices agreed by the team are adhered to.

○ The described role is explicitly defined and noticeably perceived.

○ Rituals defined in the team are guided and adhered to.

○ Any problems, obstacles or discrepancies are addressed promptly and dealt with in the team.

○ Critical points can be addressed openly and constructively.

○ There is a trustworthy team structure in which everyone can contribute and develop their strengths and weaknesses.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

The relevant expertise for the task is available in the team multiple times.

○ The skill distribution does not dictate the prioritization of work in the team.

○ Know-how is broadly anchored and know-how transfer is carried out continuously.

○ The members of the team can perform several tasks/disciplines. They remain the specialists in their field, but acquire further skills (T-Shaped Knowledge).

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

We defined at least one security specialist in the team who is responsible for the interface to a central group security function (if available).

○ We are aware that the safety of your products/services is your responsibility.

○ The security specialist is defined in the team and has been communicated to Group Security as an interface.

○ He/she has the necessary knowledge of security, DevOps and agility to identify and communicate security concerns / problems.

○ He/she ensures for each iteration that security aspects are discussed and considered.

○ All team members actively support the Security Champion and jointly contribute to the security of the product/service.

○ All team members have a minimum understanding of security aspects.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Our team works together stably over a longer period of time and we all devote most of our working time to team tasks.

○ All team members work with at least 80% of their workload in the team.

○ Our team consists of 5 to 11 people (incl. PO and Scrum Master).

○ Our team works together in a stable manner for a period of at least one year.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Our team is responsible for the product and service development throughout its entire life cycle.

○ Development, testing, deployment, operation and phase-out are carried out by our team.

○ Operations tasks are automated whenever possible

○ Operations tasks are performed by the whole team

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Team Fitness

Operation

We make realistic plans on which stakeholders can rely on.

○ We know our efficiency per iteration and consider it during the planning.

○ We inform stakeholders proactively and transparently if there are delays or deviations in planning.

○ Measures to optimise predictability are identified and implemented.

Rating

○	○	○	○	○	○
0	1	2	3	4	5

The implementation takes place in short iterations or as a continuous flow into production.

○ We introduce changes continuously or in short, synchronized iterations.

○ Each iteration generates a value for our company and/or the customer.

○ Per iteration, at least one product increment with the required quality can potentially be productively implemented.

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Our prioritized backlog can be viewed transparently.

○ We work with (exactly) one backlog.

○ The backlog is an ordered list.

○ The entries in the backlog are sorted

○ The status of the backlog item is current.

○ The backlog is transparent and accessible to all relevant stakeholders.

○ The backlog shows what can be expected with the next iteration.

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Ecosystem

We are aware of dependencies on and to other teams and actively manage them.

○ We are aware of the dependencies we have on other teams and organisations (e.g. customer care, infrastructure, suppliers, central end to end monitoring, etc.) and how our work affects others.

○ Regular contacts with these teams are established and institutionalized to effectively manage our dependencies.

○ Our collaboration with other teams is regularly discussed and possible impediments are identified.

○ Impediments are proactively discussed with the affected team and measures are derived and implemented.

Rating

○	○	○	○	○	○
0	1	2	3	4	5

To ensure that superiors and stakeholders live the principles of the Collaboration Charter, we proactively give them feedback:

○ We give our superiors and stakeholders both positive and negative feedback on the Collaboration Charter in a timely, open, honest and constructive manner.

○ We take a constructive and active approach to the conduct and activities of our superiors and stakeholders that hinder or restrict us.

○ We provide timely, open, honest and constructive negative feedback to our managers and stakeholders regarding the Collaboration Charter (we don't let anything burn).

Rating

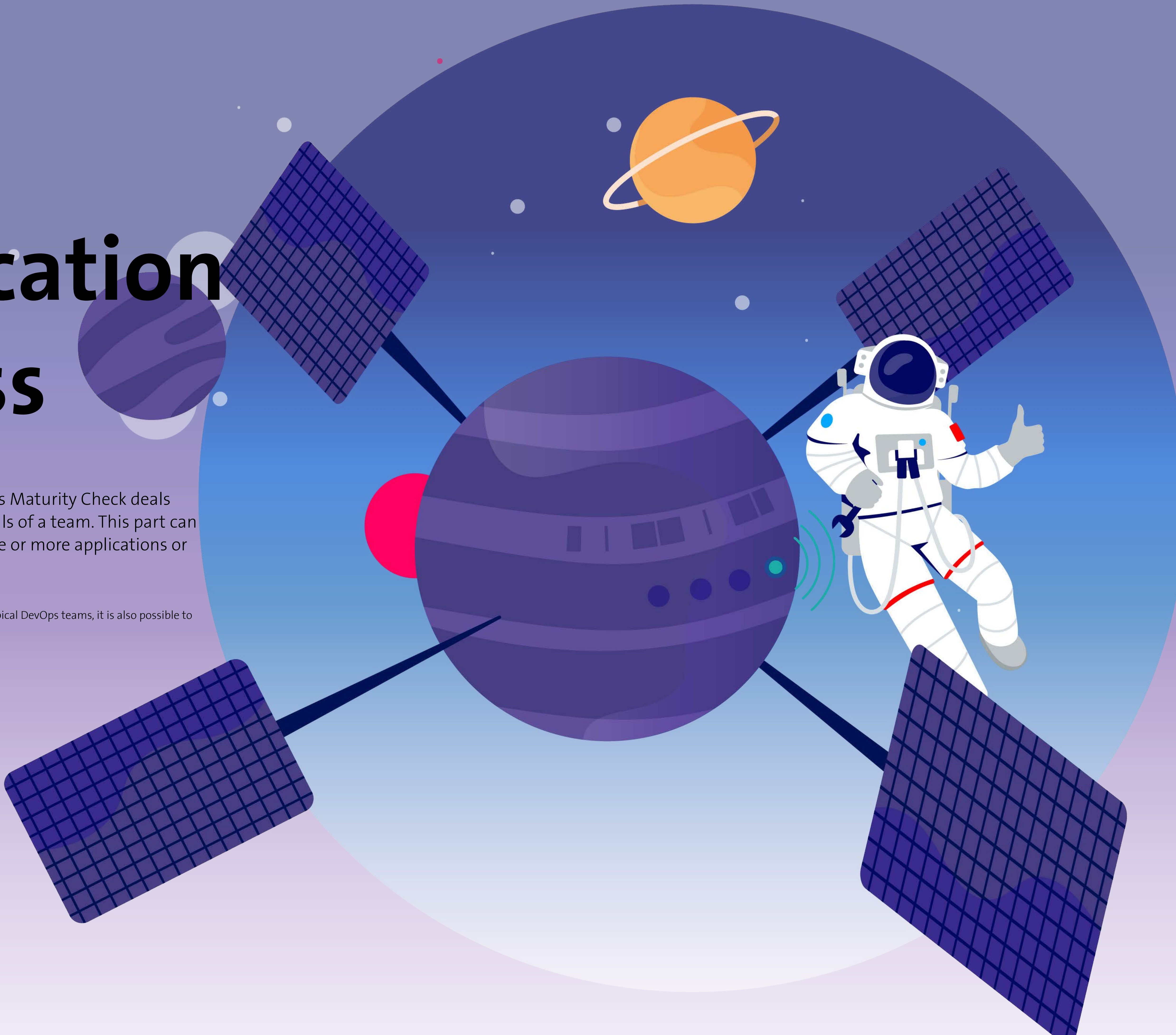
○	○	○	○	○	○
0	1	2	3	4	5

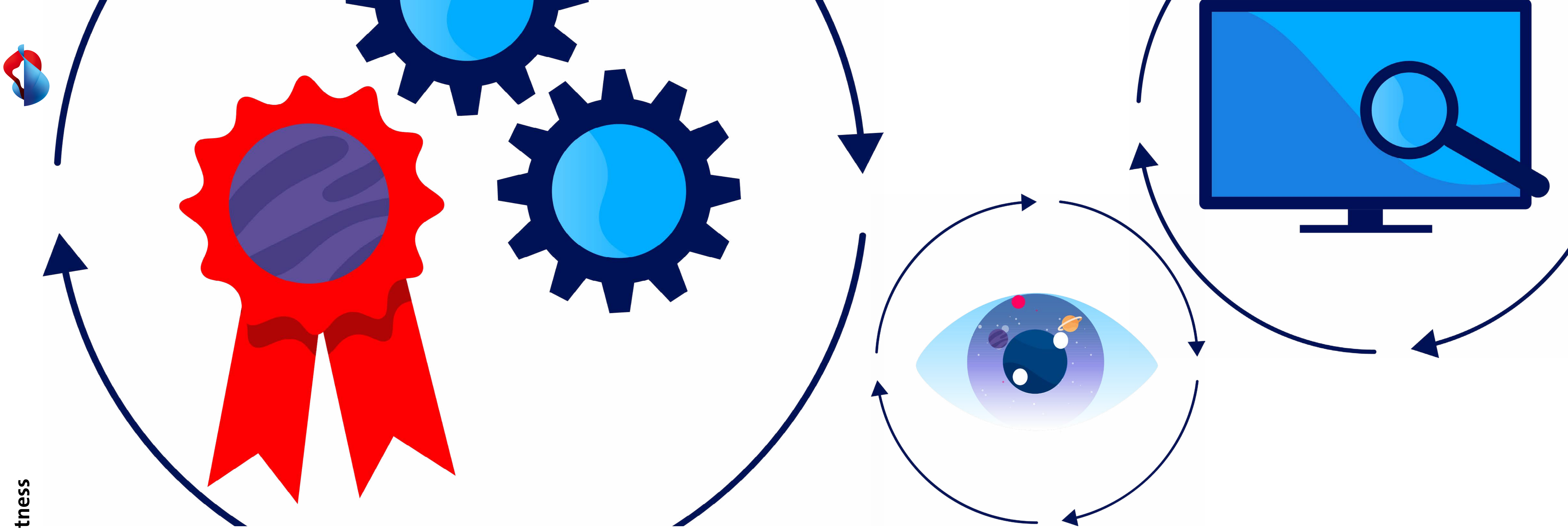
swisscom

Application Fitness

The second part of the DevOps Maturity Check deals with the technical DevOps skills of a team. This part can be performed per team for one or more applications or services.

Note: Due to the fact that not all teams are typical DevOps teams, it is also possible to skip this part.





Collaborative Developement

Highest Engineering Standards enable innovative, high quality delivery of good customer experience. It changes the way companies build applications and improves business results. Customer Experience Design is involved from the beginning. An agile development platform facilitates cross-team engagement and business feedback, enabling rapid prototyping, rapid learning and test-driven development. Application Programming Interface (API) activation technology accelerates the analysis, implementation and testing of APIs to quickly integrate new project systems into data systems. Accelerator and mobile backend services (e.g. workflow, security or push notifications) also shorten the time from concept to code and thus to the expected business value.

Continuous Testing

Highly automated approaches signicantly improve speed through a „shift left“ technique. This method provides an online service catalogue in a common portal that creates automated test workflows, combined with advanced analysis and metrics to deliver results-based testing. Continuous testing removes bottlenecks in testing through the use of virtual services and simplifies the creation of virtual test environments that are easy to implement, share and update as systems change. These capabilities can reduce the cost of providing and maintaining test environments. They shorten test cycle times by enabling integration testing earlier in the life cycle. Overall the quality of the test results increase.

Continuous Release & Deployment

Continuous releasing creates a pipeline that automates the deployment of test and production environments. Deployments at the push of a button reduce manual work, waiting times and any follow-up work due to any human misbehavior. This enables higher frequencies, fewer errors and continuous transparency.

Continuous Monitoring

A user-friendly monitoring of all platforms - not only the production environment, helps agile teams to understand the performance and availability of their application. Early feedback is critical to reduce the cost of errors and detect issues as soon as possible in the process.

Continuous Customer Feedback and Optimization

Agility is the ability, to adopt quickly. A fast feedback loop directly from the customer and an appropriate reaction to the feedback is important. Only when this is taken into account can the benefits of agility be fully exploited.

Collaborative Development

Application Fitness

Continuous Integration is implemented. Code is checked into the repository at least daily, which triggers a number of automatic processes (build, deploy, test)

- ○ Everyone can see the results of the latest build
- ○ Test in a clone of the production environment (Important: A clone of the prod still needs to fulfill the prod‘s required protection level. The handled data does not magically change its confidentiality level)
- ○ Keep the build fast (a couple of minutes)
- ○ Every commit (to baseline) should be built
- ○ Everyone commits to the baseline every day

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Automated deployment

- ○ Our source code is in a managed Version-Controlled System
- ○ All relevant deliverables are stored on a manged Repository
- ○ Each build is self-tested. All tests should run to confirm that it behaves as expected
- ○ All successfull builds are stored on a manged Repository
- ○ Automate the build with central build services

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Best practices for the development process are bindingly defined in the team and are adhered to by all. The team works in the context of „Built-in Quality“

- ○ Compliance with standards is automatically verified when the code is checked in
- ○ Code reviews are carried out for all changes or are developed using peer programming

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

In each iteration the Threat Model is updated to the latest version

- ○ Coding and naming standards are defined and adhered to
- ○ Tool-based task and defect management statuses correspond to reality at all times. All tools are linked and their status is automatically updated
- ○ All (software) dependencies are known, transparent for all and are actively managed
- ○ The documentation is updated in each iteration and always describes the current state
- ○ A Threat Model exists

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Application Fitness

Collaborative Development

The requirements were divided and defined on the basis of capabilities, features and stories. All elements are clearly defined and comprehensibly described for all involved (Procedure according to the Built-in Quality Strategy)

○ In a scaled environment, requirements are structured hierarchically (e.g. in Program Epics, Features and Stories)

○ Requirements are developed together with all relevant stakeholders, for example using Behaviour Driven Development (BDD).

○ Realized backlog items are validated together with all relevant stakeholders, for example within Sprint Demo, System Demo, Solution Demo, User Acceptance Tests, etc.

○ The Backlog Items are cut small enough so that they can be implemented within a useful period of time (e.g. an iteration)

○ We have a Definition of Ready, a Definition of Done and work with Acceptance Criteria‘s.

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Scenarios for all capabilities, features and stories are defined. The scenarios were developed in cooperation with all stakeholders.

○ For each Backlog Item scenarios are defined

○ The scenarios were defined in collaboration with the relevant stakeholder

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Continuous Testing

The test automation (White-Box and Grey-Box Testing) is available at the latest simultaneously with the finished code

○ Regression tests are managed, so that they cover the current range of functions and can be swiftly executed

○ System integration tests are derived from the acceptance criteria of the Feature

○ At the latest with the finished code fully automatic unit testing, unit integration testing, system tests, and system integration tests are at your disposal

○ Automatic security checks (e.g. security unit tests, SAST, vulnerability scans) are integrated into the development process

○ Unit tests flow into an automated regression test, which is performed automatically with the build

○ Unit integration tests are derived from the acceptance criteria of the Story

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

All tests are performed fully automated after the build. The test automation and execution is the responsibility of the team

○ Continuous automation of the entire test chain (unit, system, security and performance tests). Only a few explorative tests (< 5 %) are still performed manually

○ The responsibility for test execution is clearly defined by those involved in the large solution (acceptance criteria of Capabilities), program (acceptance criteria of Features) and team level (acceptance criteria of Stories).

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Test scenarios and data are versioned and test results can be recreated with any version at a later time

○ Versioning of the test attributes is linked to the code version

○ Tests of a certain version can be repeated at any time and produce the same results

○ All attributes relevant for the test (test data, test scripts, test results ...) are versioned

Rating

○ 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Continous Deployment

Application Fitness

Any version can be made available through self-service on any platform. If required, a separate infrastructure (VMs, DBs, Container, etc) is created with the deployment

- ○ Deployments of any version can be easily performed in self-service mode by the team and all relevant stakeholders. The installation is performed automatically
- ○ The application runs on cloud infrastructure
- ○ Required infrastructure can be created together with the deployment
- ○ All deployment scripts on all environments are fully automated
- ○ Deployment to any environment is performed with the same deployment automation

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Deployments take place without noticeable effects or downtime during the running operation

- ○ It is possible to make deployments only available for a specific group of users (e.g. with A/B Testing or Canary Releasing)
- ○ A final switch-over to the new version is only made when minimal quality tests have been carried out in production
- ○ All components are deployed without interrupting the service (blue/green deployment)
- ○ Due to the low impact, deployments can be carried out during office hours

Rating

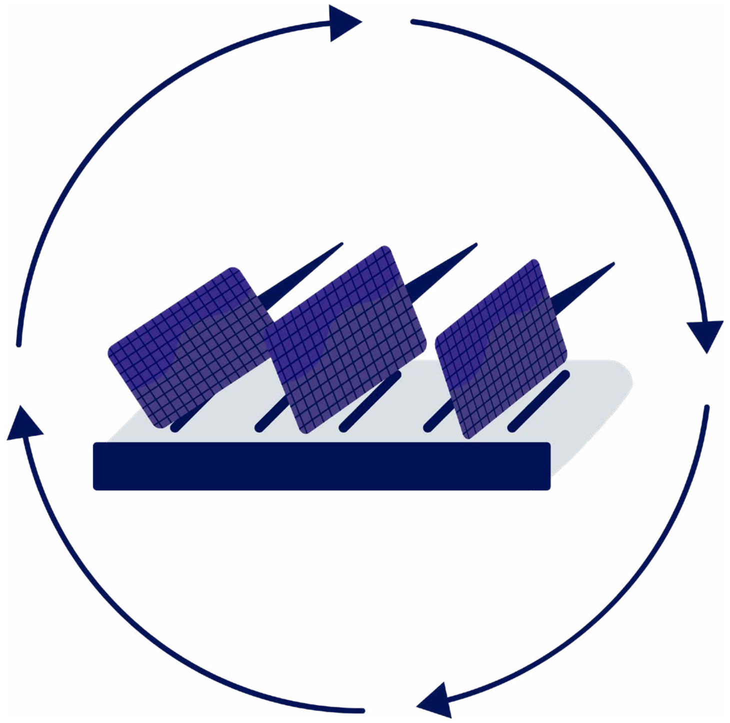
○	○	○	○	○	○
0	1	2	3	4	5

The provisioning of releases (making new functionality available on the market) can take place completely independently of the actual deployment into production (importing a new release into production)

- ○ It is possible, with Feature Toggles or Dark Launching, to place a feature on the production system, but still prevent its productive use
- ○ Any interfaces offered to other applications are backward compatible. Changes can therefore be implemented independently of each other

Rating

○	○	○	○	○	○
0	1	2	3	4	5



Continous Monitoring

Application Fitness

The relevant production monitoring information is available to all involved parties continuously and at any time

- ○ It is defined whether the service needs to be monitored from a security perspective and which monitoring information is forwarded via the defined interfaces
- ○ The central end to end service monitoring clarifies whether the service is monitored there and which monitoring information for event, incident and change management is forwarded via the defined interfaces
- ○ Monitoring information is forwarded to other involved parties/ systems (e.g. Data Lake)
- ○ Continuous and uninterrupted collection and delivery of the relevant monitoring informations from production to the relevant interfaces
- ○ The team has the relevant metrics about their services, as well as about the application of the product. The metrics are visualized and are present in the team
- ○ Security-relevant monitoring cases are derived from the metrics
- ○ The monitoring tools used are based on the defined target architecture

Rating

○	○	○	○	○	○
0	1	2	3	4	5

The monitoring scenarios are continuously developed, tested and brought into production by the team as part of the service/ product

- ○ To ensure quality, the team defines the relevant monitoring scenarios based on business and security input, knowledge of business processes and technical implementation
- ○ Monitoring is based on efficient, joint monitoring solution
- ○ The monitoring scenarios are continuously developed, tested, deployed and maintained by the team together with the service/product

Rating

○	○	○	○	○	○
0	1	2	3	4	5

The monitoring takes place on all environments (test, integration, staging, PreProd and Prod, as well as clones thereof) in order to recognize impacts of a change on the service or the monitoring as early as possible

- ○ Identical metrics are measured on all environments
- ○ The monitoring scenarios on the different environments are identical and are automatically rolled out on the environments
- ○ A negative impact of a change is detected and corrected as early as possible before going live, which has a positive effect on the Incident after Change Rate
- ○ Real-time metrics from monitoring are already used during development for control and quality assurance (Metric-Driven Development).
- ○ Metrics on all environments are transparent, are actively monitored and are available to all relevant stakeholders

Rating

○	○	○	○	○	○
0	1	2	3	4	5

Continuous Customer Feedback & Optimization

Application Fitness

Hypotheses on the expected customer behavior of the implemented change are systematically tested in production

○ This hypothesis is validated automatically and regularly with active tests in production

○ Each feature is based on a hypothesis regarding the impact (customer buys product or uses functionality, usability becomes better, errors can be corrected, performance increase, ...).

Rating

○ ○ ○ ○ ○ ○
0 1 2 3 4 5

Channels, through which customer feedback is obtained regularly and systematically are established and institutionalized

○ It is defined how feedback from the customer can be collected (this can be surveys, CUC messages, tweets, etc.)

○ Feedback is collected continuously and as automated as possible at least on a monthly basis

○ A defined and lived process exists by means of which this feedback is systematically collected

Rating

○ ○ ○ ○ ○ ○
0 1 2 3 4 5

New developments or corrections are first released for a limited group of end users (whitelist, friendly user, ...) and this group is then continuously expanded

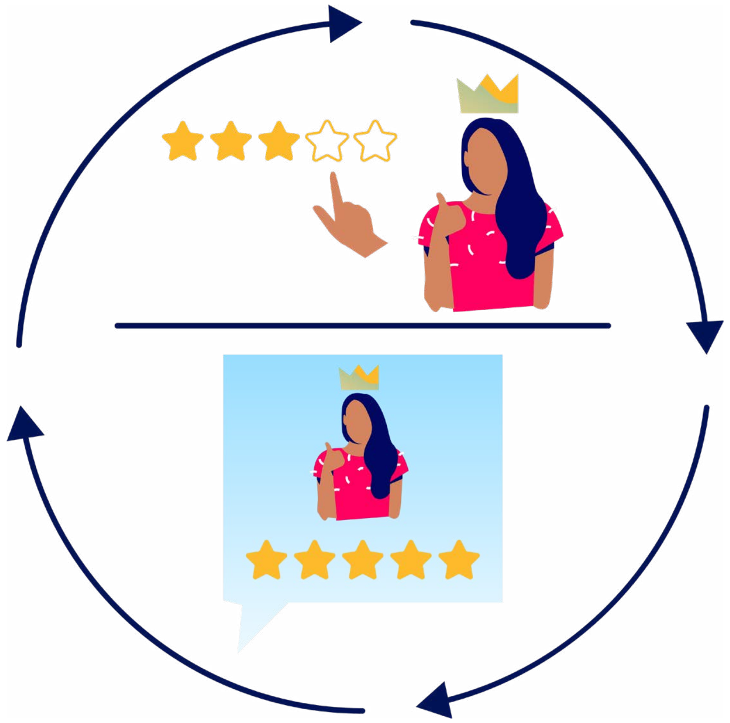
○ Each feature is first made available to a small group and automatic tests are performed

○ Only if the tests carried out are successful, the feature is continuously made available to a larger group

○ Canary Releasing and/or similar concepts are implemented where appropriate

Rating

○ ○ ○ ○ ○ ○
0 1 2 3 4 5



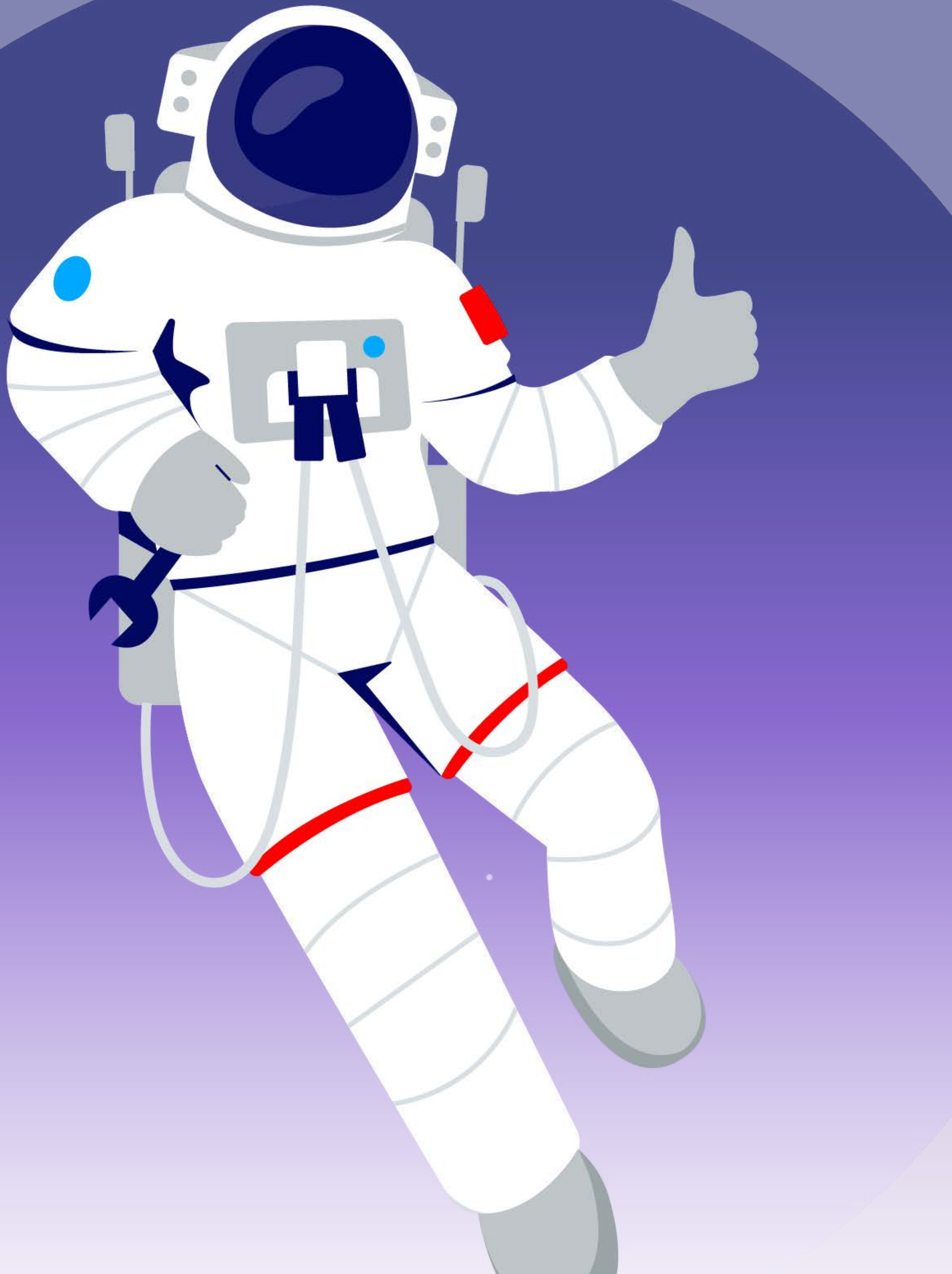
swisscom

Thank You!

Thank you for using the DevOps Maturity Check! We encourage you to tweak it to fit your team or company‘s context.

We‘re happy to hear of your experience, so please don‘t hesitate to reach out to us and give us feedback!

Good Luck!



swisscom