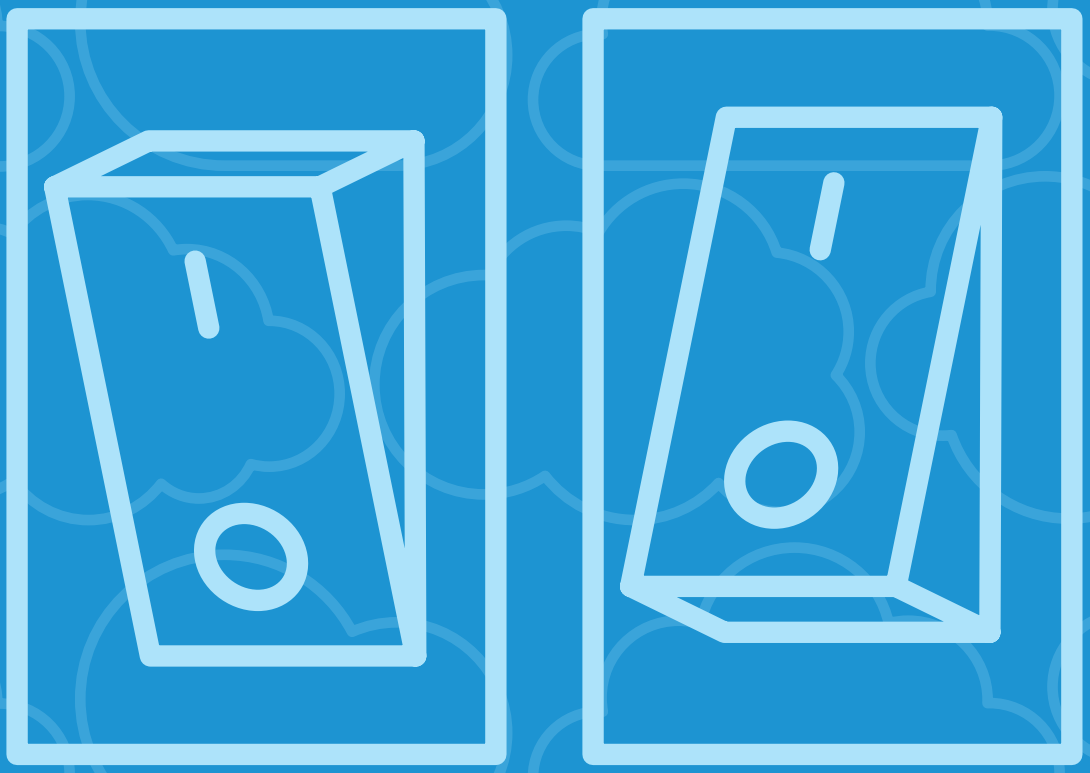
## Environment as a Service:

# A Buyer's Guide for Scaling DevOps



Quali

# Introduction

Digital transformation dictates automating processes and adopting self-service approaches that facilitate fast innovation while enabling governance.

Organizations that develop applications share the challenge of providing teams with application **environments** to support their development, test, and deployment-to-production. But application environments require infrastructure, and infrastructure provisioning often becomes a bottleneck.

Turning cloud infrastructure into a commodity that can be consumed by the **value stream** is a priority for any organization that seeks operational excellence in the digital world. Ideally, cloud infrastructure should be consumed seamlessly like power, water, or internet connectivity.

While virtualization and container technologies get us closer to this vision, actually connecting these technologies to business needs requires overcoming cultural and technological barriers.

**Environment as a Service (EaaS)** solutions help users define applications together with their infrastructure and data requirements, and make them accessible and mobile, so they can be consumed seamlessly by any process. The aim of EaaS is to eliminate the application environment bottleneck to enable faster innovation at scale.

As organizations establish their DevOps toolchain, they need to choose the right EaaS approach for connecting infrastructure to their value stream. This choice is made against a backdrop of multiple strategic decisions: the level of investment in private, public, or hybrid cloud, choice of cloud providers, the roles and responsibilities of CIO and CTO organizations, the level of investment in container technologies, serverless, native cloud services, and more.

This guide provides a comprehensive review of factors that should be considered when selecting an EaaS strategy and evaluating tools.

# Table of Contents


A. Categories

1. Cloud Choice
2. Tool Maturity for DevOps
3. Production Environment Automation
4. Self-Service Environments
5. Overseeing Environment Consumption
6. Enterprise Scale

B. EaaS Assessment Matrix

C. About Quali


## Who is This Guide For?

IT/Ops and engineering professionals, Site Reliability Engineers (SRE), and DevOps managers that are evaluating EaaS as part of their DevOps strategy.

## How to Use This Guide

The guide is divided into six categories; each category covers a set of related assessment factors. Each factor can be scored based on the factor's importance to your business. Then give each tool that you evaluate a score based on the tool's ability to deliver on that category and factor.

At the end of each section in this guide, you will find an assessment matrix with categories and factors that correspond to that section. The complete assessment matrix for all six categories can be found at the end of the guide.

# How to Use the Assessment Matrix

### STEP 1: GAUGE IMPORTANCE

Decide how important each factor is to you and your business needs. You can find more information on each factor throughout this guide. When scoring, use the range depicted below.

0 - Not Important  1 - Somewhat Important  2 - High Importance

The importance score you decide on, should be kept the same for all evaluated tools.

### STEP 2: EVALUATE TOOLS

Evaluate each tool's ability to deliver on each factor in all six categories. Pay attention to how they deliver value to the factors with the highest impact to your business.

### STEP 3: SCORE TOOL'S VISION

Score each one of the tools that you are evaluating based on the tool's vision and ability to provide value for each factor. Use the scoring range depicted below when filling this out.

0 – No Vision  1 – Partial Vision  2 – Complete Vision

### STEP 4: CALCULATE

Once the full assessment matrix has been filled in, calculate each tool's total score by adding the products of each factor's importance score by the tool's factor score.

$$\text{Total Tool Score} = \sum_{(i=1)}^{18} (\text{Importance Score}_i \times \text{Tool Factor Score}_i)$$

After each tool's total score is calculated, compare each tool's total score to help guide your decision on scaling DevOps with an Environment as a Service solution.

Building a Complete Environment as a Service Solution:

# 1. Cloud Choice

Your business strategy and organizational characteristics dictate your choice of cloud infrastructure.

# 3 Questions to Determine Your Cloud Choice

### *1. Do you plan to support on-prem infrastructure side-by-side with public infrastructure?*

Many organizations have existing CAPEX investment in on-premise infrastructure, however within the organization, there may be teams that don't align on the investment's future. For some organizations, it is strategically important for them to own and continue investing in their on-premise infrastructure.

Others may wish to focus on phasing out their on-premise infrastructure.

Keep in mind that continued investment in on-prem infrastructure may limit your EaaS choices. If this is a critical factor for your business, make sure you filter out approaches that don't support hybrid cloud.

At the very least, you should weigh the burden of managing separate solutions for on-prem and public cloud infrastructure.

### *2. Are you in a position to commit to a single cloud provider?*

Even with a cloud-first strategy and a plan to eventually migrate the majority of workloads to public cloud, you still need to evaluate the simplicity of being a single-cloud-provider shop against the price and strategic risk of vendor lock-in.

Most cloud providers offer their own toolchains and services, which could make your life easier (not just for EaaS). However, it comes at the price of potentially making it very difficult to change course later, when new requirements and technologies are introduced.

**Cloud choice options may be important if:**

- Your organization supports teams in multiple geographies
- You need to support a variety of tasks and workloads that may be optimized on different cloud providers
- You need to adhere to multiple compliance standards
- Cost and SLA optimization is a high priority

Different cloud providers excel in different areas, and if your organization size or field mandates a best-of-breed approach, cloud-agnostic solutions need to be taken into consideration.

# 3 Questions to Determine Your Cloud Choice

### *3. Do you need support for multiple technologies?*

Are you 100% container-based? Do you use native cloud services? Do you deploy your application on cloud instances? Whatever your cloud composition is, you may be in the process of planning to evaluate additional technologies. As you evaluate your EaaS approach, your vendor's ability to support multiple technologies should be taken into consideration. Developing a separate strategy for cloud instances and containers could make it difficult to provide a consistent service to your **end users**.

## Cloud Choice Assessment Factors

Blending your cloud investment can help reduce risk and costs. If this factor is meaningful for your organization, you may need to put higher weight on this set of criteria

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| Environment blueprints support multiple cloud providers | | | | |
| Native support for cloud instances, cloud services, and containers | | | | |

# 2. Tool Maturity for DevOps

The level of DevOps maturity for the EaaS tools you select is critical for your ability to effectively build your toolchain. While most evaluated tools claim “DevOps support”, it is essential to drill down and understand the nature of such support.

# 3 Levels of DevOps Tool Maturity

### *Level 1: Integration with DevOps ecosystem*

At the most basic level, any new tool must integrate with the DevOps ecosystem including your CI/CD pipeline tools, source control tools, artifact repositories, secret stores, test automation tools, security tools, logging and monitoring tools, and more. While exposing a full REST API to facilitate integrations is a must, specific plug-ins that make the integration seamless are quickly becoming a standard expectation.

### *Level 2: Horizontalization of tools*

Innovation is a continuous process that has no beginning nor end. It therefore requires a standard toolchain that connects our work silos into a flow.

The second level of tool maturity is the continuity of tools. Continuity makes it possible to use tools across the value stream. These tools do not live in any specific stage.

Test automation, for example, is no longer an activity that happens between development and release. It could run anywhere from development to production, probably in all of them. Testing in production is an approach that is gaining traction. Today's test automation tools no longer tie into a specific testing phase, but are built to provide continuous testing services to the value stream. This makes it possible for QA professionals to establish their role in DevOps and take responsibility for quality throughout the value stream, instead of retaining their traditional "gatekeeper-to-release" role.

Similarly, EaaS cannot be limited to any specific stage of the value stream. This leads to continuity of infrastructure, and the desire to provide a holistic approach to environments, all the way from early planning and development to production.

### *Level 3: Automation reuse*

While the second level of maturity indicates that the same tool would be able to provide services throughout the value stream, the third level of maturity requires these tools to be aware of the value stream, and therefore encourage smart reuse of automation throughout the process. Automation reuse is key in seeing return on automation investment. This requires more sophisticated and process-aware tooling.

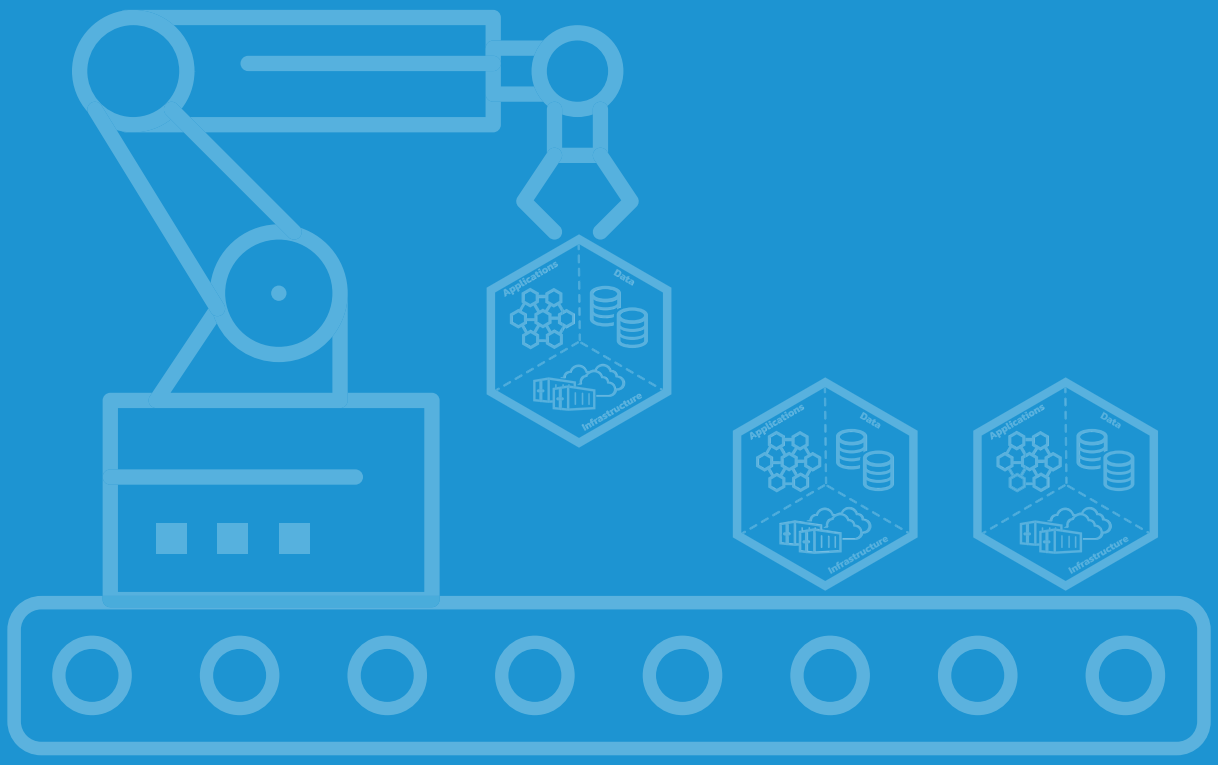
We can consider the maturity of classic Infrastructure-as-Code tools, for example. While the same tool can theoretically provide application environment automation in the different stages of the value stream, it typically does not model applications. Because applications connect to the business process and drive reusability, it's crucial to be able to model applications effectively. Reusing these tools at scale may require building an additional level of abstraction, which could be difficult and costly if done in-house. You may not see the need immediately, but it will become apparent as your business scales and requires a higher level of DevOps maturity.

# 3 Levels of DevOps Tool Maturity

## Tool Maturity for DevOps Assessment Factors

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| Easy to provision and deploy environments automatically throughout value stream via strong integration with DevOps ecosystem tools such as value stream management and CI/CD pipeline tools, source control, artifact repositories | | | | |
| One environment solution across the value stream | | | | |
| Smart **blueprint** reuse between pipeline stages by abstracting applications from infrastructure | | | | |

# 3. Production Environment Automation




When adopting a holistic approach to EaaS throughout the value stream, the same automation should be used for all environments, from development to production. This section covers the key considerations for automating your production environment.

# Key Considerations for Automating Your Production Environment

### *Supporting production environments automation*

If you use a certain automation to set up a dev/test environment, this automation is also validated with any test that runs on it. It makes sense to use this process to gain confidence in the environment automation, so it can be taken into production.

However, using automation tools that weren't designed for production environments would require in-house development of missing capabilities to address the limitations. Examples for such gaps would be Access Control and Production Environment Isolation.

### *Isolation of Environments vs Reuse*

Introducing automation that can be reused in pre-production and production should not mean that production must become the developers' responsibility. Traditional principles of security, compliance, and auditability should be applied to EaaS. Different EaaS approaches would provide varying levels of support for these principles, and whatever is not supported out-of-box would have to be completed or ignored—depending on the organizational strategy and needs.

Application Release Automation (ARA) tools provide a traditional approach to release automation that narrows Production environments to one silo in the value stream. In that, they limit the reuse of automation. While this siloed approach offers the advantage of isolation, this may be weighed against reuse and optimization depending on use case.

### *Deployment strategies*

Adopting the concept of **immutable infrastructure** makes EaaS simpler. It facilitates making all environments dynamic, including what is traditionally the most static of environments--Production. With immutable infrastructure, high maintenance change management is replaced with simple instantiation of new production environments and switching from the old environment to the new one, from infrastructure provisioning all the way to application deployments. This opens the door to more advanced deployment strategies, including blue-green and canary. When implemented effectively, EaaS makes it possible to reuse the dev/test environments automation, used for testing a new version, to deploy the same new version as a "green" production environment. This is an element that must be taken into consideration when assessing potential approaches.

# Key Considerations for Automating Your Production Environment

## Production Environment Automation Assessment Factors

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| Dedicated support for provisioning and deployment of production environments | | | | |
| Isolation of pre-production from production environments (blueprints, infrastructure and data) | | | | |
| Support for production deployment strategies (blue-green) | | | | |

# 4. Self-Service Environments


One of the complexities of the continuity of tools in DevOps is that it mandates considering the needs of multiple personas. The stakeholders throughout the value stream often have conflicting agendas, and therefore providing a holistic solution for their challenges is a balancing game.

Coders are not the only consumers of environments. Other environment end-users include developers and engineers, QA, IT/Ops, DevOps, SRE, security, support, product, documentation, and training team members.

Understanding the end users' preference in consuming EaaS is imperative when vetting out approaches.

# Establishing EaaS for The Entire Team

### *Balancing ease of use throughout value stream*

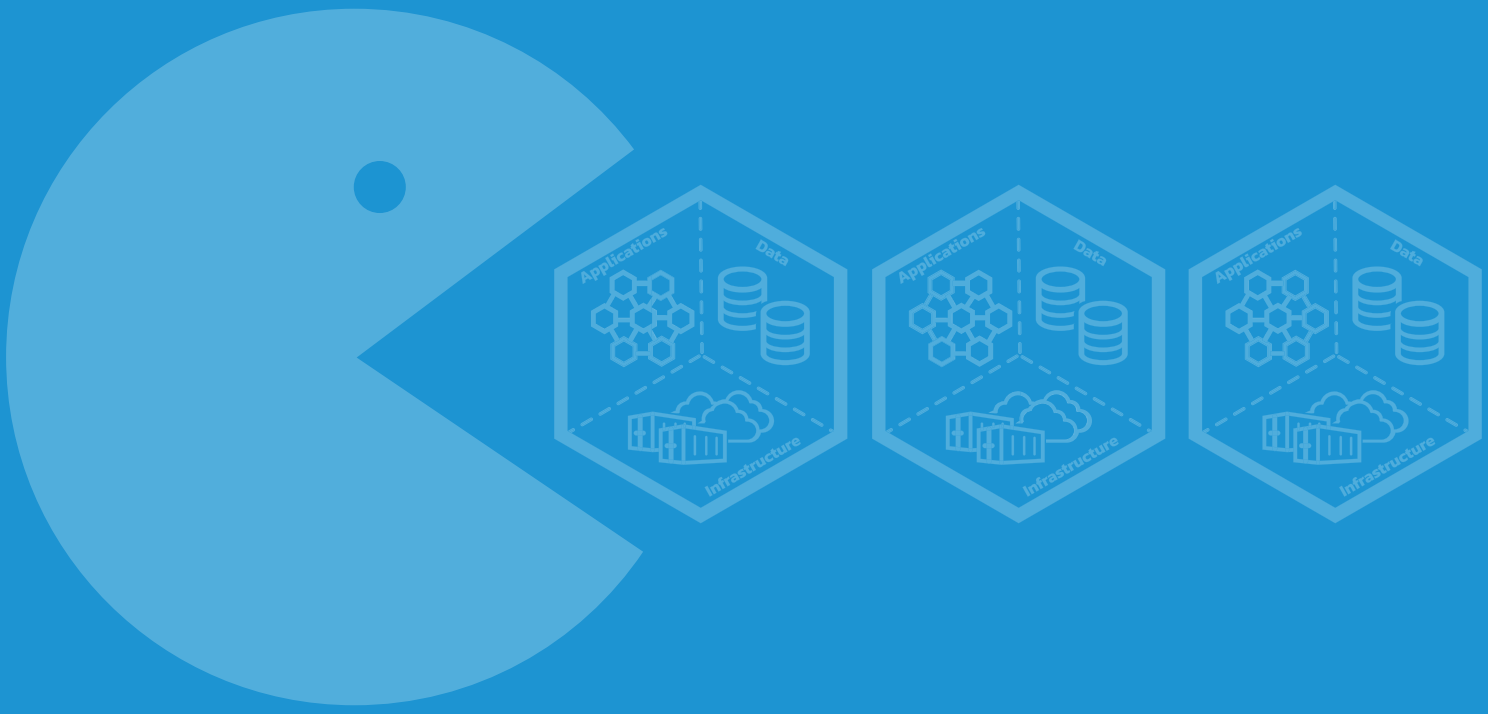
Developers who need environments may want to consume them via command line or API. They may prefer ChatOps tools like Slack, Hipchat, or Teams. Other users may prefer non-code or low-code consumption models. Integrating with ChatOps and providing a UI-based self-service is usually required. Such users would likely find text-file environment representation unreadable and would demand visual environment representation that highlights important information and parameters. They would require the option to select environments from a friendly catalog that offers a familiar search experience and other navigation. These users may be internal to the organization or potentially external (including, for example, contractors who perform periodical penetration testing).

Adopting an inclusive EaaS approach is important regardless of the coding skills of the team that provides environments. Including the end-users of environments in the decision process, or at least collecting their requirements, is key in establishing the best EaaS approach.

### Self-Service Environments Assessment Factors

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| Easy to provision and deploy environments via self-service by developers, developer friendly interfaces (command line, 100% API coverage) | | | | |
| Easy to provision and deploy environments via self-service by non-developers (testing, ext. testing, planning) | | | | |

# 5. Overseeing Environment Consumption




DevOps started from developers, but to be adopted successfully it must empower other teams. While Developers may thrive on OSS and unlimited flexibility, other teams may not find this world as appealing.

For automation to scale in the DevOps value stream, non-experts must be able to use the automation tools.

# Creating Cross-functional Environment Consumption

### *Teams responsible for providing environments*

The objective of the team responsible for providing environments is to increase innovation speed. These teams usually have a core of strong coding skills, and the ability to explore and integrate OSS tools. The teams usually want to leverage assets that were already created with OSS tools. This would have to be taken into consideration. Multiple teams may have different tooling preferences, and this may require either developing integrations or opting for an EaaS solution that is open enough to integrate with multiple tools.

### *Teams responsible for controlling cost, access, security, and compliance*

One objective of these teams is to reduce risk. While teams that provide environments would opt for any tool that makes automation easier and faster, the teams responsible for cost, security, and compliance need to make sure that the proliferation of automation doesn't let things get out of control.

An example of this is public cloud costs. Environment automation may increase the speed of application development and release, but costs must also be considered. It must be possible to map cloud costs to environment consumption. This can be done by **tagging cloud resources** in environments with metadata that can later be extracted from the cloud provider and be used to track cloud resource usage and connect it to cost and business value. Therefore, teams responsible for cloud costs should require that automatic tagging is made part of the EaaS strategy.

Similarly, they should require that automatic de-commissioning of ephemeral environments is enforced (for example dev/test environments). No one wants to pay for public cloud resources that go unused, but in fact, this waste is a large driver of unnecessary public cloud expense. These requirements should be considered to ensure that the selected EaaS approach gives you the right combination of operational speed and cost control.

# Creating Cross-functional Environment Consumption

## *Overseeing Environment Consumption Assessment Factors*

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| Built-in integration with open-source infrastructure-as-code tools | | | | |
| Analytics on infrastructure cost for pre-production and production environments | | | | |
| Policies restrict environment pre-production usage | | | | |
| Automatic de-commissioning of dev/test environments | | | | |
| Automatic tagging of public cloud resources | | | | |

# 6. Enterprise scale


Any approach evaluated by an enterprise organization would require considering enterprise readiness. Required capabilities include Role Base Access Control (RBAC) on Environments, the ability to streamline security standards in the EaaS process, and the auditability of the process.

# Preparing for Enterprise Adoption of EaaS

## *SSO and RBAC*

Examples for SSO and RBAC aspects:

- General support for SSO via multiple identity providers and multi-factor authentication
- The ability to define different roles for EaaS administrators and end-users
- The ability to define which environment blueprints different end-users can see and use
- The ability to define which cloud providers are available for each group of end-users

## *Security*

Security aspects would have to be specific to organizational needs, and may include the following:

- Avoiding exposing end-users to cloud provider credentials
- Secret management in environments
- Secure pre-production environments – avoiding exposure of public IPs, automating https certificates

## *Auditability*

Auditability may include:

- Ability to track historical Production deployment information
- Recording usage of environments throughout the value stream

## *Enterprise scale Assessment Factors*

| Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|
| SSO and RBAC | | | | |
| Security | | | | |
| Auditability | | | | |

# EaaS Assessment Matrix

**How to Score:**
Determine the weight of importance. Then compare the different tools and add the total.

| # | Category | Importance | Tool 1 Score | Tool 2 Score | Tool 3 Score |
|---|---|---|---|---|---|
| 1. | **Cloud Choice** | | | | |
| | Environment blueprints support multiple cloud providers | | | | |
| | Native support for cloud instances, cloud services and containers | | | | |
| 2. | **Tool maturity for DevOps** | | | | |
| | Easy to provision and deploy environments automatically throughout value stream via strong integration with DevOps ecosystem tools such as Value Stream Management and CI/CD Pipeline tools, Source Control, Artifact Repositories | | | | |
| | One environment solution across the value stream | | | | |
| | Smart blueprint reuse between pipeline stages by abstracting applications from infrastructure | | | | |
| 3. | **Production Environment Automation** | | | | |
| | Dedicated support for provisioning and deployment of production environments | | | | |
| | Isolation of pre-production from production environments (blueprints, infrastructure and data) | | | | |
| | Support for production deployment strategies (blue-green) | | | | |
| 4. | **Self-Service environments** | | | | |
| | Easy to provision and deploy environments via self-service by developers, Developer friendly interfaces (command line, 100% API coverage) | | | | |
| | Easy to provision and deploy environments via self-service by non-developers (testing, ext. testing, planning) | | | | |
| 5. | **Overseeing Environment Consumption** | | | | |
| | Built-in integration with open-source infrastructure-as-code tools | | | | |
| | Analytics on infrastructure cost for pre-production and production environments | | | | |
| | Policies restrict environnent pre-production usage | | | | |
| | Automatic de-commissioning of dev/test environments | | | | |
| | Automatic tagging of public cloud resources | | | | |
| 6. | **Enterprise scale** | | | | |
| | SSO and RBAC | | | | |
| | Security | | | | |
| | Auditability | | | | |
| | **TOTAL** | | | | |

Quali
Environment as a Service

# Glossary

**Blueprint** - An environment definition that includes information on the planned applications, data, and infrastrcture. Blueprints promote reusing the same set of definitions to set-up multiple environments.

**Environment** - A set of applications and their data that need to be deployed on provisioned infrastructure resources. Environments may be needed for any stage of the value stream, including production, development, testing, staging, troubleshooting, support, and more.

**End Users** - Members of teams that consume environments for DevOps tasks—from developers to test automation engineers, QA, security, Ops, IT, and more.

**Environment as a Service (EaaS)** - Defines applications together with their infrastructure and data requirements, and makes them accessible and mobile, so business processes can consume them seamlessly. The aim of EaaS is to eliminate the application environment bottleneck to enable faster innovation at scale.

**Immutable Infrastructure** - An approach that replaces application upgrades and configuration changes on static infrastructure with instantiation of a new environment and switching from the old environment to the new one—from infrastructure provisioning all the way to application deployment.

**Tagging cloud resources** - Tags are metadata that can be added to cloud resources that are consumed from public cloud providers like AWS and Azure. This metadata can later be extracted from the cloud provider and be used to track cloud resource usage and connect it to cost and business value.

**Value Stream** - aka the DevOps Toolchain. The sequence of steps required to satisfy a need.

# About Quali

Quali provides the leading platform for Infrastructure Automation at Scale. Global 2000 enterprises and innovators everywhere rely on Quali's award-winning CloudShell platform to create self-service, on-demand automation solutions that increase engineering productivity, cut cloud costs, and optimize infrastructure utilization.

# Contact:


**NORTH AMERICAN HQ**

10801-2 North MoPac Expressway
Suite 200
Austin, TX 78759, USA

+1 877 QUALI 10
+1 408 588 1774
info@quali.com

**ISRAEL**

7 HaPsagot St ,
Building B floor 5
Petach-Tikva 4951000, Israel

+972 77 901 4000
info@quali.com

**CALIFORNIA**

4353 N 1st St.
Floors 1 & 2,
San Jose, CA 95134, USA

+1 877 QUALI 10
+1 408 588 1774
info@quali.com


Quali
Environment as a Service

Information in this document is accurate as of 12/01/2020. Information may have changed after this date. 120120_v1