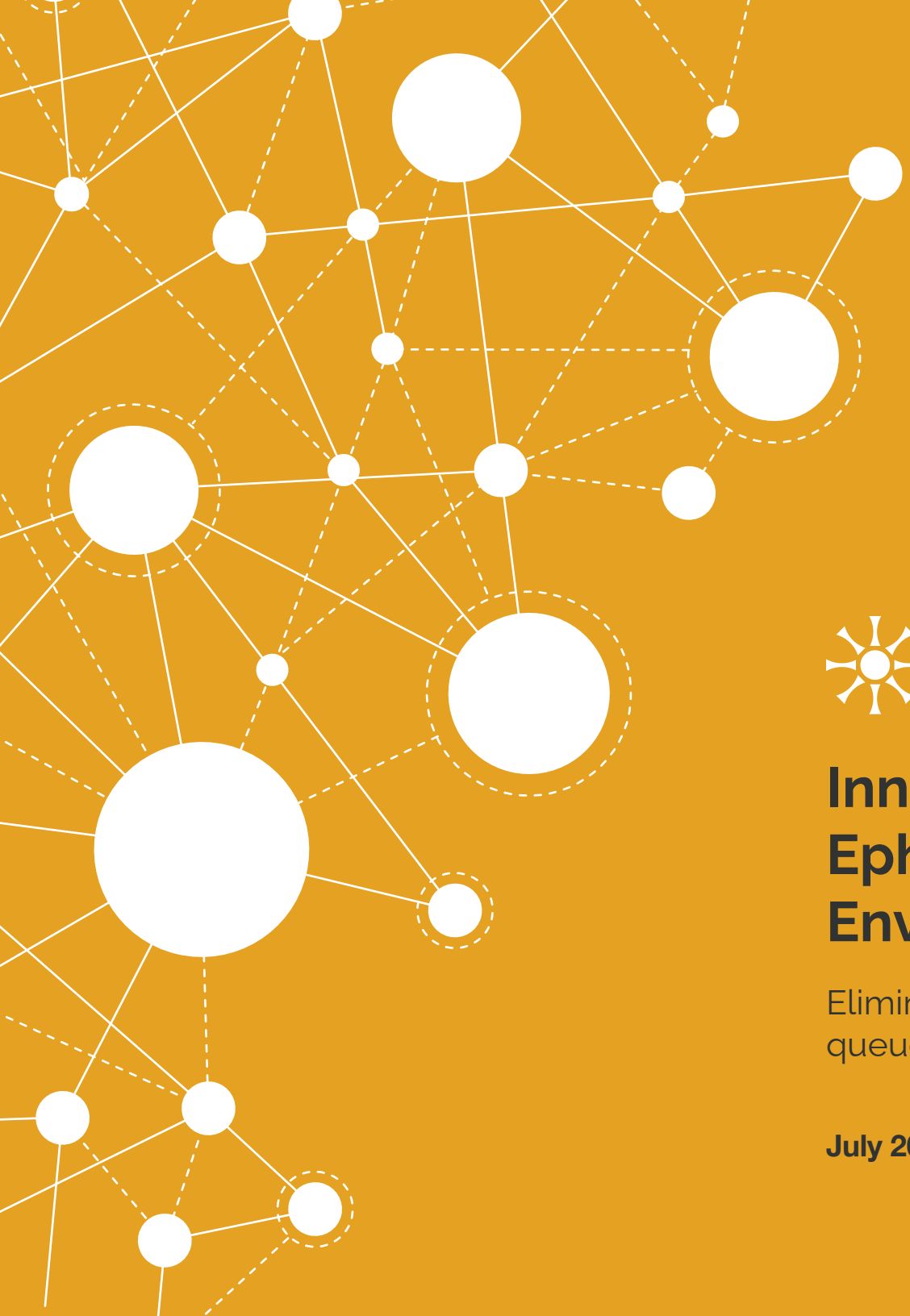


roost.ai

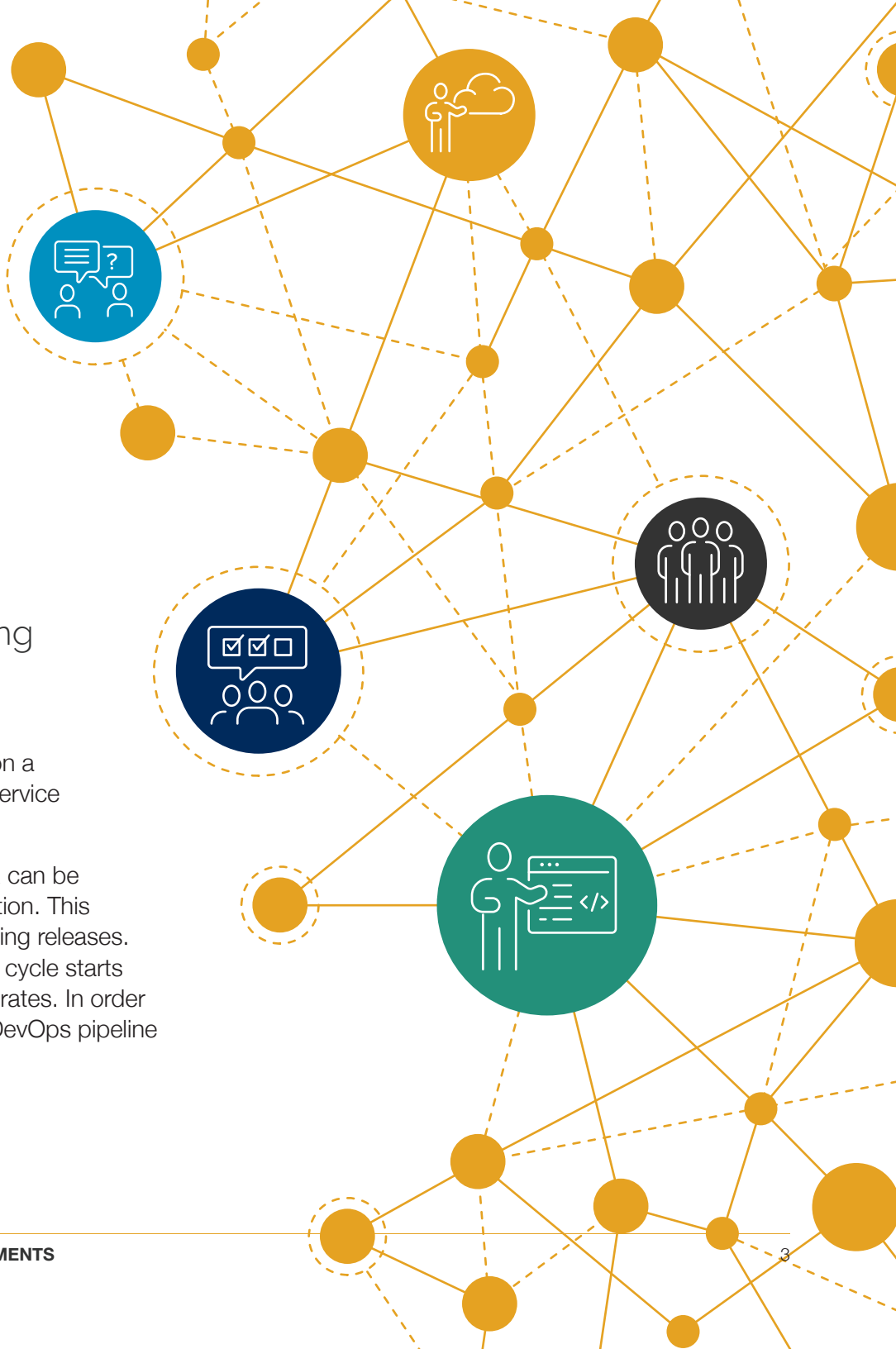
Innovate Quickly with Ephemeral Pre-production Environments

Eliminate static staging sites and testing queues with automation

July 2022

Contents

Ephemeral Pre-production Environments: The New Gold Standard for DevOps



It is time to rethink the software development life cycle and eliminate static staging and testing queues in order to innovate more quickly.

With the number of services proliferating, developers are now working on a smaller and smaller subset of them. This has made effective testing of service changes and pull requests (PR) with dependencies a monumental task.

Current processes require developers to wait in a queue before their PR can be tested or is accessible to a staging environment for testing and certification. This process slows down PRs from merging into the main branch thus delaying releases. And, to make matters worse, if the pull request fails in testing the whole cycle starts again — consuming resources and time while increasing change failure rates. In order to maximize the productivity gain of a pull request, every artifact in the DevOps pipeline needs to become ephemeral.

What is an “Environment as a Service”

An environment as a service (EaaS) is a space where teams can experience and interact with shareable and testable applications crafted using configuration, infrastructure and dependent services.

EaaS is a way of delivering development infrastructure and services to run an application in an ephemeral testing space that is both dynamic and shareable. It removes the burden of creating custom scripts and managing complex software in order for developers to create a pre-production testing environment.

What Does Ephemeral Mean in Terms of Technology

Ephemeral simply means without fixed boundaries on life spans. An ephemeral environment is used as long as it is needed, no more and no less. The primary difference between an ephemeral environment and a long-lasting environment is not necessarily the length of time it runs, but that the ephemeral environment can be quickly spun up and just as easily discarded.

It also means there are clear stages of life. At the start of life and during lifetime, an entity can borrow resources from the environment and when it shuts down or terminates, these resources are (hopefully) returned back to the environment.

Ephemeral environments used for software development are designed fundamentally to be short-lived and to eventually go away.

How Ephemeral Environments Benefit DevOps Teams?

Ephemeral environments speed-up releases by automatically testing an application with only relevant dependencies which are auto-provisioned at pull request.

Ephemeral environments are a powerful and simple way to on-demand spin up the most complicated pre-production environment. These preview environments can be created for every pull request, developers can work on multiple features at the same time and simultaneously share environments with stakeholders for review. This increases deployment velocity, eliminates change failure rates and speeds up the release of containerized microservices decreasing downtime in production.

is the expected increase in frequency of software releases within the next two years.

"Organizations are under more pressure than ever to drive faster innovation and deliver new digital experiences to their customers, partners, and employees. In response to this, DevOps and SRE practices are becoming increasingly critical. Organizations have already made great progress driving DevOps across their applications, but need to scale these efforts further to deliver new products and services with maximum speed, quality, and reliability." [1]

1 *https://assets.dynatrace.com/en/docs/report/13488-global-devops-report-2021.pdf*

Obstacles Found in Current DevOps Process

Manual and Antiquated Processes Inhibit Speed of Innovation

In the *2021 Global DevOps Report*, Dynatrace cites that DevOps teams are expected to increase the frequency of software release by 58 percent within the next two years.

No software development teams can increase release rates this much without modernizing their current manual development processes that are done in isolation or testing that occurs on a static staging site.

Teams that rely on custom scripting only create drag and make changes slow and labor intensive.

Without modernization of the way environments are created, managed and discarded innovation speeds will continue to be inhibited and the rate of innovation will not meet demand.

A Bloated Software Development Life Cycle

In the past two decades, a five-stage software development life cycle has become the standard process. However, it over-emphasizes the three middle stages. The two most important phases are development and production.

Development is where the application comes to life, while production is where it actually lives. Engineers need to focus on these two stages – the beginning and the end – to move applications quickly and efficiently to production.

But the problem lies in the fact that, in today's development process, there is no place for developers to test and validate code with production-ready services. Also, there are issues with traditional testing environments which only allow developers to do basic testing and validation of code. It becomes impossible for efficient testing to occur when dealing with modern complex architectures such as containers, micro services and cloud-native applications. Complexity scales faster than the number of services.

The three-phase, pre-production stages (continuous integration), traditionally helped with unit testing of applications. This worked well for monolithic applications, but with the proliferation of containers and micro services has lost its effectiveness. With the size of code continuously shrinking, micro services and applications have now become first-class citizens. A new framework is needed to battle harden these services and cloud-native applications which focus on service-to-service issues.

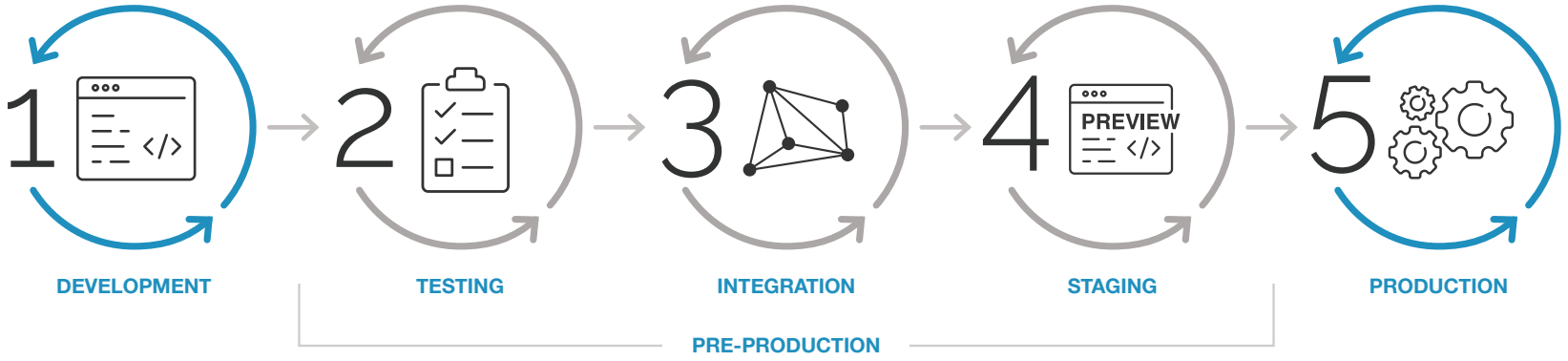


Figure 1. Traditional Five-step Software Development Cycle

Traditional Static Testing Methods Aren't Enough

Testing is an integral part of any DevOps release cycle and many continuous integration (CI) tools have "run tests" as one of the software development processes. These processes do basic testing and validation of code; however, the static nature of most testing sites makes it impossible for efficient testing to occur when dealing with modern complex architectures such as containers, micro services and cloud-native applications.

And, to complicate matters, every pull request has a nuanced definition of the environment it needs, and may have requirements not provided on most test environments.

Even in a staging environment there may not be access to the application's production infrastructure, dependencies, services, databases, etc. – so real-world testing against a pre-production environment can not actually be performed.

Because of the static nature of testing environments, they will never meet the demand of evolved artifacts and versions of services required by a pull request.

And then there are the wait times for the hand-off from development to Q/A bogging down the process even more with queues for testing and review slowing down the speed of innovation to a crawl. According to the *2021 State of DevOps Report* (page 7), *"58 percent report that multiple handoffs between teams are required for deployment of products and services."*

Rethinking the SDLC Phases

"It makes sense to have an intermediate step between development and production to make sure the application has a chance to be evaluated by someone other than the original developer before going into production. But there's no reason this step should require three separate environments — one intermediate environment should be enough. I think a good name for this environment is "production-like," making the software development life cycle; development –> production-like –> production." [2]

THENEWSTACK

Lack of Collaboration

"Successful DevOps efforts require collaboration with all stakeholders. More often than not, DevOps efforts are instead limited to infrastructure and operations (I&O). Organizations can not improve their time to value through uncoordinated groups or those focusing on I&O exclusively." [3]

Gartner

"Works on My Machine" Issues

Software developers often work in isolation and manually set up and configure workspaces on desktop environments or services that are isolated in the cloud. These static environments can fall out of synchronization with each other and lead to different dependency versions and bugs that are due to environmental inconsistencies and not necessarily code issues.

Current development processes do not provide an effective methodology for engineers to test code or have it certified for release before it is pushed to the next stage of the cycle. Developing in isolation and in static environments is the root cause of "works on my machine" issues. Because developers can not effectively test against real production assets, bugs and integration issues are pushed into the next phase causing downtime and hindering timely release cycles.

The article, *The Secret to DevOps Success,*[4] points out that the lack of collaboration is a key issue to why DevOps teams face failure.

2 Toward a 3-Stage Development Lifecycle, *https://thenewstack.io/toward-a-3-stage-software-development-lifecycle/*

3 *Gartner, Inc., The Secret to DevOps Success, April 2019, Katie Costello, https://www.gartner.com/smarterwithgartner/the-secret-to-devops-success*

4 *2021 State of DevOps Report*, page 35, *https://puppet.com/resources/report/2021-state-of-devops-report*

"The most highly evolved organizations in our DevOps models are adopting a platform model that enables self-service for developers and curates the developer experience… A highly effective platform provides a guided experience for the customers of the platform and that platform is treated as a product. It enables stream-aligned team members to focus on the things most important for their customers and get common building blocks and tools from the platform. Its purpose is to ensure delivery is smoother and faster." [5]

puppet by Perforce

5 *2021 State of DevOps Report*, *Page 35, https://puppet.com/resources/report/2021-state-of-devops-report*

Three Ways to Streamline DevOps Processes and Increase the Speed of Innovation

Environments as a Service platforms allow teams to realize the core goals of digital transformation: increased deployment velocity, decreased downtime in production and more creative uses of software throughout the organization.

1

Accelerate Release Cycles while Reducing Cloud Costs

EaaS platforms auto-discover environment configuration by inspecting source-code repositories and automatically test and validate code changes. They are a powerful and simple way to spin up the most complicated pre-production environment.

They remove the burden, cost and development time of hand-crafting scripts and managing complex software in order for developers to create a pre-production environment for sharing with stakeholders and automated testing. Eliminate the cost of environments running 24/7 by scheduling the deployment time.

2

Static Test Environments Become Dynamic Pre-Production Environments

The only way to ensure a pull request is completely accurate and ready to merge into production is to test the PR in a pre-production environment that mirrors the entire application. Hand-crafting these environments for every PR would be impossible, and at the end of the day they are still static.

An EaaS environment by design is dynamic and runs “test cases” automatically and “learns” from system behavior so it can run dependency tests in a much faster and more efficient way. When a PR is merged (or a change moves to the next step) it’s already validated and ready for production. This eliminates the need for another test environment.

With EaaS platforms developers can test service changes automatically and continuously with dependent services and production configuration within the development phase of the cycle. This eliminates the need for CI in a unique and left-shifted way and removes many of the unnecessary steps between development and production.

Since the environment is defined by a pull request and private to the developer numerous PRs can be run in parallel. No more QA bottlenecks!

Working Together

"To break down barriers and forge a team-like atmosphere… varying teams must work together, rather than in uncoordinated silos, to optimize work." [5]

Gartner

5 *Gartner, Inc., The Secret to DevOps Success, April 2019, Katie Costello, https://www.gartner.com/smarterwithgartner/the-secret-to-devops-success*

3

Stakeholders become Part of the Application Development Process

An ephemeral preview environment can be shared simultaneously with each stakeholder via a custom URL. They can validate deliverables using their own criteria, ensuring that each and every service is certified prior to being released to production.

Cycles can be shortened because the traditional testing, integration and staging are no longer needed — they have shifted into one pre-production stage.

In short, stakeholders become a part of the review and development process right from the start and no longer have to wait to see the change in production.

Figure 2. A three-stage software development life cycle accelerates development velocity and pull request merge success.

A Modern Tool for to Improve Developer Velocity and Reduce Cloud Costs

Roost Ephemeral Environments as a Service Platform

Roost is an Environments as a Service (EaaS) platform that on-demand creates an ephemeral pre-production environment. Engineers can effortlessly spin up an ephemeral environment on-demand, at a specific time, or at every pull request. Once created, a Roost preview environment URL can be shared with stakeholders to validate their deliverable. This eliminates custom script creation and managing complex software in order to develop in a pre-production environment.

The whole release cycle gets accelerated using this fast feedback loop, accelerating development velocity and pull request merge success.

The Roost platform

- **Effortlessly spin-up a pre-production environment** at every pull request, feature branch, or insertion point in the DevOps / GitOps pipeline.
- **Auto-discovers environment configuration** by inspecting source-code repositories (e.g. GitHub, GitLab, BitBucket) and optimizes it using the power of machine learning.
- **Automatically tests and validates** by accessing all necessary containers and micro services required to test and validate code changes instantly.
- **Reduces costs from environments running 24/7** because they can be spun up on-demand then automatically destroyed upon PR merge.

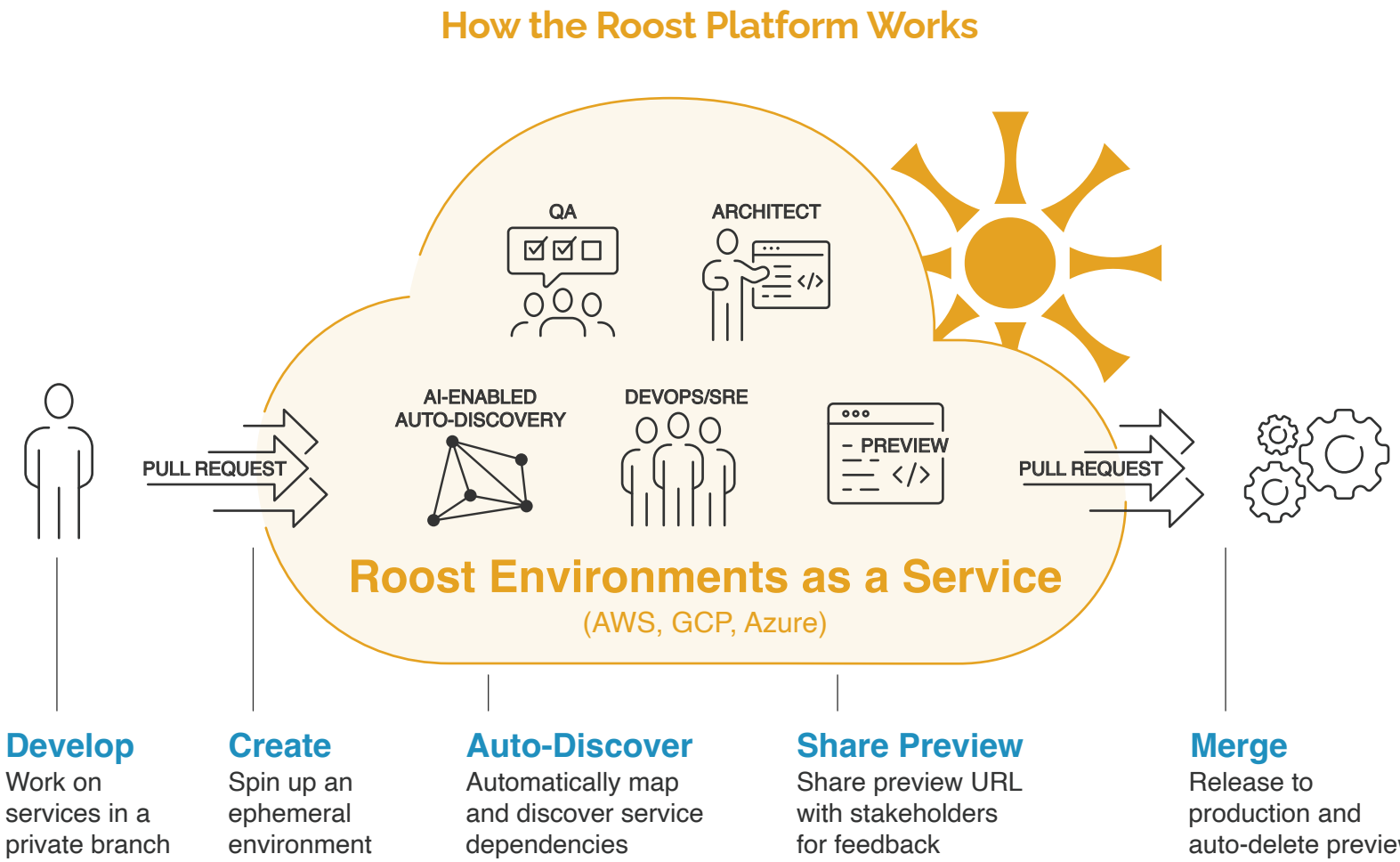


Figure 3. Roost Environments as a Service Platform helps accelerate PR merge success rates by removing the traditional CI phase of the SDLC.

About Roost

Roost is an Environments as a Service platform that creates an ephemeral pre-production environment on-demand or at every pull request. By inspecting source-code repositories the platform auto-discovers and maps environment configuration then optimizes it using machine learning. Once created, a Roost preview environment URL can automatically be shared with stakeholders. The whole release cycle gets shorter using this fast feedback loop accelerating development velocity and pull request merge success.

roost.ai

roost.ai | 4950 Hamilton Avenue | San Jose, CA 95130

070122