@ZeroStackInc sales@zerostack.com www.zerostack.com

# Clearing the IT Ops Pipeline: Removing Obstacles to Agile Software Development from the Continuous Integration/Continuous Delivery and Deployment Cycle

White Paper

As the demand for rapid delivery in digital services grows, software companies are looking for standardized ways to release products and upgrades in weeks instead of months, or in days instead of weeks. Many providers are now adopting the Continuous Integration / Continuous Delivery and Deployment (CI/CD) cycle of agile software development to address the high velocity of releases and faster time-to-market requirements. In the CI/CD cycle, developers and testers continuously integrate and test new code, deliver applications and upgrades to production, and from there, deploy them to customers through cloud or web-based venues.

But old problems of software development still remain. In most companies, Dev/Test teams are dependent on IT and Operations (IT Ops) teams to provide the physical and software stacks they need to continuously produce and release software and upgrades. But IT Ops teams can't provide developers and testers with infrastructure and applications fast enough to meet the ongoing and relentless pace of the CI/CD cycle. In short, developers "need it now," and IT Operations can't get it to them quickly enough.

Developers, testers, QA, and IT Ops teams face certain obstacles that create bottlenecks in the IT Ops pipeline, slowing down the software development cycle. In this white paper, we'll examine those obstacles and ways to remove them. Our goal is to accelerate the end-to-end CI/CD cycle by helping IT Ops teams deliver IT infrastructure and software tools to developers in a shorter time frame. We'll also look at how a standardized cloud-based platform of provisioning and procurement tools – a common need for many software providers – can assist Dev-Test and QA teams in their goals of Continuous Delivery and Continuous Deployment.

## Obstacles to Faster Development Speed

The obstacles that cause bottlenecks in the IT Ops pipeline are often due to old habits and obsolete tools in software organizations. The most common IT Ops obstacles are:

### Lengthy Procurement and Provisioning Cycles

Often, developers don't make requests to IT Ops for new physical infrastructure and software all at once, but piecemeal, as they need certain items. One week, they might request "a server with three databases, with Oracle Database and MySQL installed on it;" the next week, they might request "six VMs;" and the next week, they might request "a Jenkins server, with Ansible and RabbitMQ installed."

> Lengthy procurement and provisioning cycles can result in developers waiting anywhere from several days to a few weeks for IT Ops to handle each request.

In many software companies, the IT Ops team has a lengthy procurement and provisioning cycle, and developers may have to wait anywhere from several days to a few weeks for IT Ops to handle each request. The reasons for the delays vary, but the most common reasons are:

- **Too many developers, too few IT Ops people** – A team of, say, 10 IT Ops people may be supporting the needs of 200 developers, testers and QA people. The higher ratio of developers means that IT Ops people are often overworked in their efforts to provide the necessary infrastructure and software for development and testing.
- **A backlog of requests** – IT Ops team members must take developer and tester requests in the order they receive them, and often must complete each request before taking on another one. This leads to a backlog. Unless they are marked "Urgent," individual

requests must sometimes sit in the queue for days or weeks before the IT Ops person can take action on them.

- **Shortage of resources** – What happens if a tester requests 20 servers for load and scale testing and QA, but IT Ops only has 10 servers available? The tester has to either scale back his/her testing plans to use the available 10 servers, or wait until IT Ops buys another 10 servers, or until 10 additional servers become available.

## Lack of Automation and Standardization

There is a lack of automation in IT Ops procurement and provisioning cycles. For example, developers may be using an outdated IT ticketing system to request infrastructure and software. Each time a developer issues a ticket request, an IT Ops person must follow a certain workflow to fulfill that request, and must complete each step before sending the ticket on to the next IT Ops person in the chain.

These workflows often consist of time-consuming manual steps. For example, an IT Admin may spend hours manually copying database software from a file server over to the hardware using Command Line Interface (CLI) scripting. The Admin must then install and configure the database software and connect it to the network. Or, if a tester or QA specialist requests a clone of the production environment, it may take an IT Ops person several days or weeks to create a duplicate of the current environment, using slow manual steps.

These manual workflows also make IT Ops prone to errors in fulfilling requests. For example, a developer may request "a server with six databases, with Oracle Database and MySQL installed on it." But when they get the server from IT Ops, they discover it only has four databases, plus MySQL has not been installed. The developer must then create and send a new ticket asking for a correction, and must wait additional days or weeks for IT Ops to correct the mistake.

Another problem is lack of standardization. Many software providers do not have a standardized platform that allows their Dev-Test and QA teams to do both Continuous Delivery and Continuous Deployment. While some advanced teams are doing Continuous Deployment, many others are struggling to get to Continuous Delivery because they don't have access to a standardized set of tools that allow them to create environments to develop, test, and deploy their applications at the speed demanded by their customers.

## Silos Within Organizations Create Obstacles

In large enterprises, IT Ops teams have silos – that is, individuals or sub-teams with different responsibilities at different levels. Each individual or sub-team fulfills certain tasks for a Dev/Test request separately from the others.

For example, in large organizations, an IT Ops team may have an Infrastructure Ops sub-team, with one person responsible for procuring and configuring servers, a second person responsible for setting up storage, a third for network connections, a fourth for security, etc. – and a separate Software Ops sub-team, where one person handles database administration, a second handles installation of workload management tools, a third handles middleware, etc.

These silos turn IT Ops procurement into a series of handoffs, where each team member must complete their assigned tasks before handing the job off to another team member. This results in delays, where DevOps people must wait for requests to move through the IT Ops pipeline as each person in each silo performs their separate tasks. The silos create a lack of collaboration between separate IT Ops teams, and between IT Ops and DevOps, when ideally, they should all be working together to accelerate the procurement process.

## The Public Cloud – Advantages and Dangers

In many companies, Dev/Test teams have turned to public clouds as a low-cost, easy-to-use solution for self-service resource provisioning. Developers and testers can set up VMs and storage on AWS and other public clouds in just a few minutes, bypassing the delays and bottlenecks of their internal IT Ops pipeline.

> "When self-service provisioning can be done quickly and consistently through complete virtualization, you eliminate the obstacles to give developers, testers, and QA the environments they need."[1]
>
> – Gene Kim

But using public clouds for the Agile CI/CD process carries its own risks. If all your developers use public cloud environments on every DevOps project, the costs can quickly skyrocket out of control. Also, it's easy to create a sprawling, undisciplined IT environment where your company isn't sure which workloads your developers have on AWS or other public clouds.

(For a complete discussion of the advantages and pitfalls of using public clouds for Dev/Test purposes, see ZeroStack's companion paper: *"A Balanced Workload: Optimizing the Cost and Performance of AWS Using a Hybrid Cloud."*)

If your company is developing applications that use intellectual property or confidential client information, or require high security or government compliance, using public clouds for DevOps may not be an option. Even if your developers are permitted to use public clouds, you may want to restrict their use of them, requiring them to use internal IT Ops resources unless they have a specific need (i.e. a two-week development deadline) to utilize public cloud environments.

## The Solution: A Self-Service Automated Private Cloud

A self-service automated private or hybrid cloud will remove the obstacles described above, helping to clear the IT Ops pipeline while providing developers, testers, and QA people with the IT infrastructure and tools they need for ongoing CI/CD development. At a minimum, a self-service automated cloud should include:

### On-Demand Self-Service Provisioning

A self-service private cloud should offer a set of automated provisioning tools to allow developers and testers to create their own Dev/Test environments. It should include both a self-service user interface (see below), and an API-driven infrastructure-as-code that lets developers create VMs and databases, access storage, set up network connections, etc., using RESTful API coding.

In essence, a self-service private cloud automates the provisioning process by letting developers and testers manage their own initial deployments and configurations. This removes many of the confusing and error-prone manual steps that IT Ops people must go through to deliver the infrastructure and software stacks that developers need. It also removes the silos within the IT Ops organization, as it puts provisioning of individual elements (VMs, databases, storage, etc.) in the hands of developers.

1. https://www.itworld.com/article/2890492/is-devops-doomed.html

## An Intelligent and Well-Organized User Interface

A self-service private cloud should have an intelligent user interface (UI) that gives developers access to a set of common infrastructure tools (i.e. VMs, Oracle or SQL Server databases, storage, network connections) and applications. This eliminates the old ticketing-based processes where users must submit multiple tickets to IT Ops to build their physical and software stacks, allowing developers to set up their own DevOps environments on the private cloud with just a few clicks.

Ideally, your self-service UI should allow your IT Ops administrators to assign resources in an organized manner. They should be able to designate Business Units (BUs) on the UI (i.e. "AppDevTeam1," "WebDevTeam1") based on units within your company, assign team members to BUs, and designate current projects (i.e. "AppDevProject1") on the UI according to BU. They should also be able to assign quotas to each BU (i.e. AppDevTeam1 gets 100 VMs, 25 vCPU cores, 100 GB memory, and 400 GB storage), and to each individual project (i.e. AppDevProject1 gets 10 VMs, 5 vCPU cores, 8 GB memory, and 40 GB storage).

## Application Store

A self-service private cloud should have an online "application store" that provides single-click access to common CI/CD tools and services, such as:

- CI/CD tools such as Jenkins, Git, Maven, and Junit
- Workload management tools such as Ansible, Puppet, and Chef
- Middleware services such as RabbitMQ and Redis
- Storage backends such as MySQL, Postgres, Cassandra, and MongoDB

NOTE: The private cloud should also allow IT Ops to place commonly-used applications on the self-service UI, to give developers and testers even easier access to them. For example, IT Ops can make a duplicate of the production environment available on the UI for testers.

> SaaS private cloud solutions have the potential to become the de-facto standardized platform for Continuous Integration and Continuous Delivery and Deployment.

They can then update the clone production environment every few months to make sure testers are using the most current and accurate duplicate.

## Seamless, two-way cloud migration

A self-service hybrid could should have seamless, two-way migration, to allow developers and testers who have public cloud access to easily move applications and workloads back and forth between public and private clouds.

## Administrative dashboards

A self-service private cloud solution should have a set of administrative dashboards that allow your IT Ops team to control access to resources on the UI. These dashboards should give you complete visibility into which teams and team members are utilizing which resources, and allow your IT Ops team to perform admin tasks, such as:

- Creating user accounts and passwords on the UI
- Creating Business Units (BUs), and projects for each BU, on the UI
- Assigning developers and testers as "members" of a certain BU or project
- Assigning self-service infrastructure tools (compute, storage, network) on the UI, and set quotas for those tools according to BU or project
- Importing new applications to the application store or the UI.

### A Software as a Service (SaaS) platform

A Software as a Service platform with portal-based access is the best venue for a self-service private cloud solution. A SaaS platform gives you the flexibility to do upgrades and customization to the various features of the UI, application store, and administrative dashboards. The flexibility of the SaaS private cloud solution gives it the potential to become the de facto standardized platform for Continuous Integration and Continuous Delivery and Deployment in Agile software development.

## Conclusion

Both Dev/Test and IT Ops teams are under pressure to support the demands of Agile development, but they have different goals. Developers and testers want to achieve a faster time to market in delivering applications to their customers. Meanwhile, IT Ops teams want to achieve a faster time to value in delivering IT resources and applications to Dev/Test teams, to support their goal of faster time to market.

As we've seen, the obstacles that hinder these goals – lengthy procurement and provisioning cycles, lack of automation, internal silos that create bottlenecks – are formidable, but can be overcome with the right technologies. A self-service, automated private cloud empowers developers and testers, giving them the tools they need to create their own DevOps environments. It also frees IT Ops teams from manual provisioning tasks, allowing them to provide developers and testers with IT resources in a more direct and timely manner. In short, a self-service private cloud helps both developers and IT Ops teams achieve their separate goals of faster time to market and time to value by clearing IT Ops obstacles to support the ongoing CI/CD cycle.