imperva

Business Continuity

Service Availability and Reliability

Imperva Cloud Security Solutions

Table of Contents

The Importance of Service Availability

Introduction

Service reliability and availability are critical to the success of your business with attacks on web applications causing increased disruption, preventing transactions, and interrupting continuity of services for your customers. The purpose of Imperva Cloud Application and Edge Security solutions is to stop these attacks and help ensure uninterrupted business operations. To achieve this goal, we have built our network and platform foundations on a resilient infrastructure designed to provide a reliable service.

This document provides insights for professionals in technical roles, such as, Chief Technology Officers (CTOs), Chief Information Security Officers (CISOs), Security Architects, Operations team members, and Security team members.

Its mission is to help you understand how Imperva Cloud Application and Edge Security is purpose-built as a fault tolerant system, designed to deliver high availability. This document also details how the operational processes implemented at Imperva enable the solution to react in a fast and effective manner to provide business continuity to users.

Imperva cloud security platform

Imperva delivers comprehensive web application, API, and network protection through its cloud platform, including several products such as Cloud WAF, CDN, Advanced Bot Management, DDoS Mitigation, and DNS Protection, all orchestrated seamlessly within a single security stack.

Our growing global network currently includes 50 Points of Presence (PoPs) around the world positioned close to our customers' clients to provide an optimal user experience.

To make good on our commitment to customers, we've engineered our cloud platform for reliability. We operate it with an SLA of 99.999% availability for CDN / Cloud WAF / DNS Protection and 100% availability for Network DDoS Mitigation services.

To get a better understanding, it is worth noting that application and infrastructure typically deployed on a public cloud environment region are delivered with a 99.99% availability objective.

To achieve such high levels of reliability, Imperva operates multiple data centers independently to considerably reduce the likelihood of two of them failing over at the same time.

When building the Imperva Cloud, we applied multiple principles to mitigate hardware failures, infrastructure failures and software bugs. These principles enabled us to build and operate high-availability systems. We will discuss these principles in greater detail later.

A reliable network

The network as a foundation

Our solutions are extraordinarily available because our network is extraordinarily reliable. All Imperva security services rely on customer traffic flowing from any point in the Internet into our PoPs (usually through local ISP services). In the PoPs, multiple network devices handle the traffic, which then flows to our single security stack. Finally, local ISPs or transit ISPs that connect to other geographical regions forward traffic to the origin website or the company IP address.

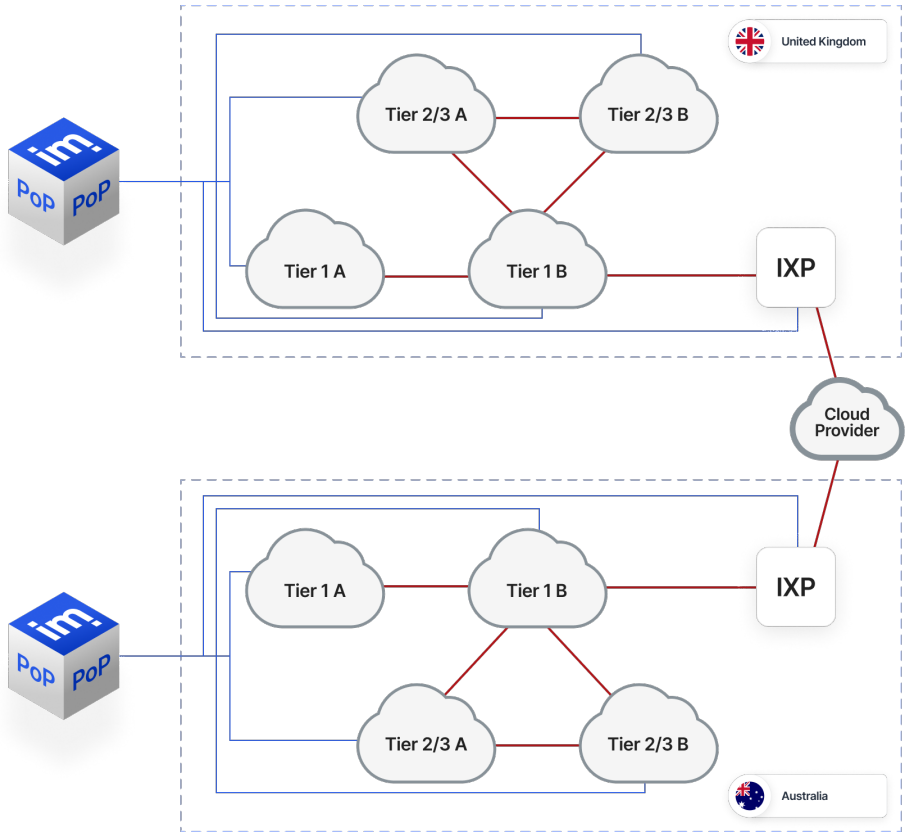
Imperva relies on dedicated teams of network engineering and network operation experts focused mainly on the network layer health to ensure our network infrastructure is as reliable as possible.

Here are some of the network design principles we follow:

The network design of each PoP is architectured for high availability. We use industry best-of-breed network devices and strive to make sure there is no single point of failure in the PoP topology.

Each PoP is connected to the Internet via different interfaces:

- Local Connectivity via multiple dedicated local ISPs and IX (Internet Exchange) for optimal in-country performance and IX Cloud providers connectivity
- Transit Connectivity via resilient global ISP strategy



- Imperva security services are exposed to the Internet using the BGP protocol. When an IP is published to a specific ISP it leads incoming traffic from the area into the PoP. This mechanism also allows our NOC (Network Operation Center) to control the incoming traffic into the PoPs and divert it to different ISPs / PoPs, simply by stopping the publication of specific IP ranges to specific internet ISP connections. By doing so, they can mitigate performance issues at the ISP connection level or even at the PoP itself.

Our NOC operates best-of-breed industry monitoring solutions, leveraging network device telemetry (e.g., flow-based monitoring, IPSLA, etc.) and advanced analytics for accelerated troubleshooting of network issues and applications performance.

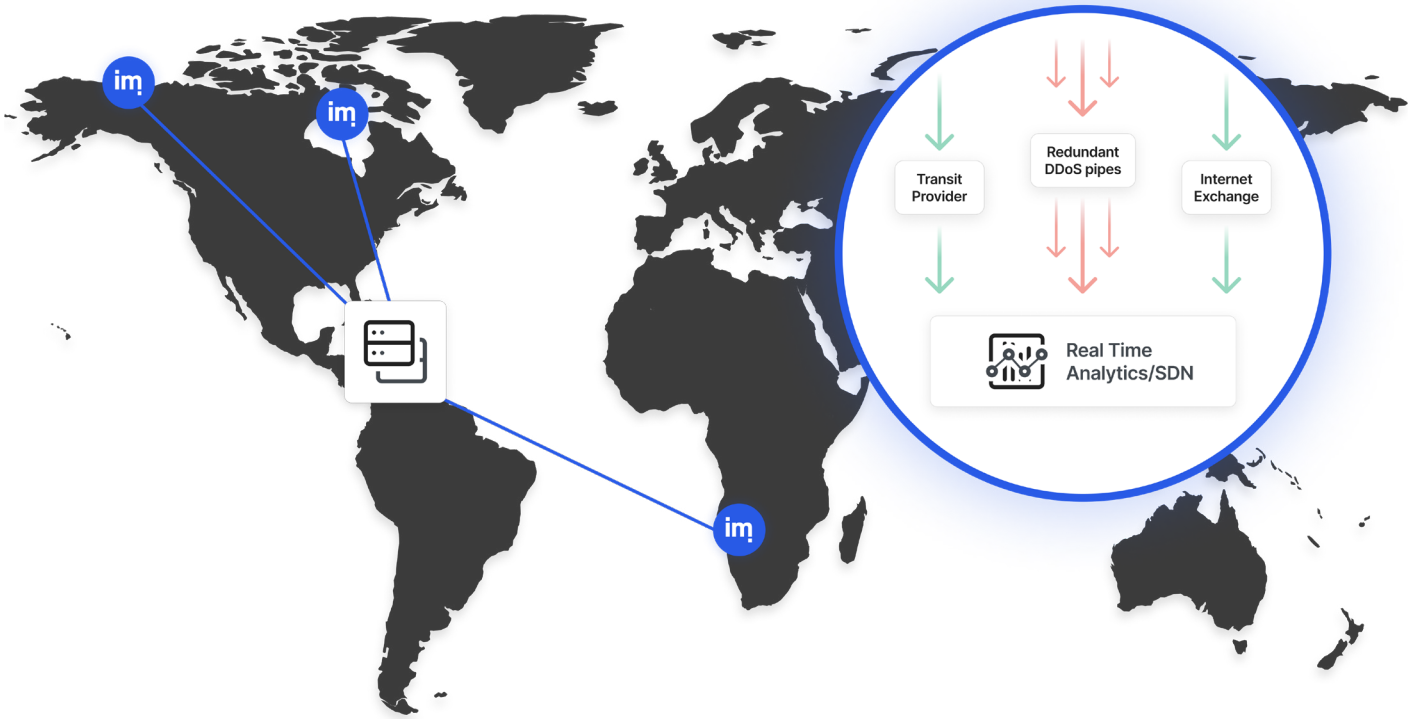
As part of our network automation strategy and by leveraging Software Defined Network (SDN) techniques, we also introduced a Software-Defined NOC platform designed to self-adapt to failures and automatically adjust network flows. This network monitors traffic congestion and other network-related failures and is designed to automatically react to these failures by attempting to mitigate the problem.

DDoS is part of the game

Volumetric DDoS attacks are growing in scale; Imperva has recently mitigated attacks exceeding 1 Tbps.

Via a combination of DNS-based steering and anycast routing, attack traffic is distributed across multiple PoPs. At each of these PoPs, our proprietary Behemoth devices are configured to identify and filter out attack traffic within seconds.

The largest volumetric attacks can overwhelm interconnection capacity between providers, however when this happens, we can quickly detect the issue and migrate traffic automatically to sites with available capacity.



Infrastructure and Platform Reliability Architecture

Physical Layer Resiliency

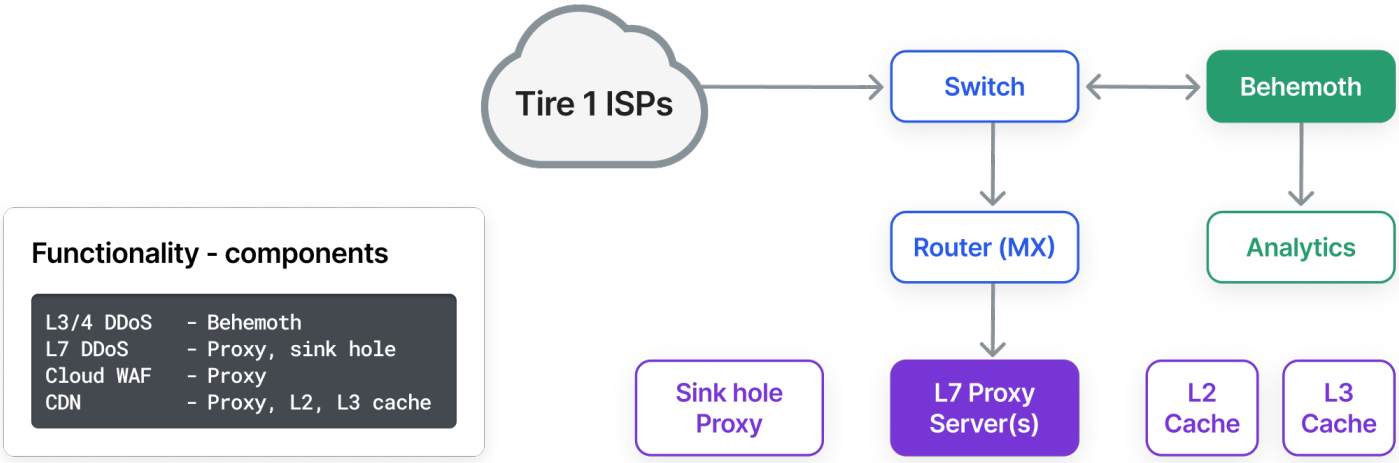
Imperva's infrastructure operates in purpose-built and standards-compliant data centers, equipped with redundant power entry, UPSes, generators, and cooling systems. Our network hardware is supplied by A/B power inputs feeding multiple power supplies, designed to allow us to operate through maintenance or failures on both device power supplies and upstream systems.

Network and Systems Resiliency

Within each data center, Imperva interconnects with multiple internet transit carriers and maintains direct peering with local networks. These connections are split across multiple routers and Behemoth DDoS mitigation devices, to avoid single points of failure.

Resiliency for WAF and CDN follows a similar model. Each WAF and cache is a part of a pool, supporting the site but carrying no unique data or configuration. As capacity requires, caches are added to the cluster. Failed hardware is replaced and software is upgraded as necessary, without customer impact.

PoP Architecturee (Real Time components)



Cloud Level: All our PoPs work together to provide service across the globe. They are placed in various regions around the world in order to be close to end users at the edge which allows us to provide best CDN performance. End user traffic can flow to any of these PoPs while using the BGP routing protocol to control the traffic flow. This way the different PoPs within a region can provide redundancy to each other. In the very rare case of PoP failure / maintenance, the other PoPs will be able to seamlessly accept and handle the additional traffic. This can also overcome public network issues.

Isolation and Fast Failover

Isolation is the system's way of identifying failures and isolating them (i.e. making sure that the failed resource is no longer being used). Once a failed component has been isolated, a failover mechanism is designed to kick in to allow the other redundant components to take over the load.

All of our nodes operate on this principle. An example of isolation is in the Cloud WAF proxy application. The software runs various self diagnostic checks including CPU and memory consumption monitoring. If it reaches the conclusion that something is wrong, such as a hardware failure or a software bug, it will crash and isolate itself, allowing for the opportunity to heal, which we discuss in the “Self healing software” section below.

Load balancing between proxy machines is carried out using a router in order to achieve maximum performance. When a proxy fails, the router is configured to sense it and forwards the incoming traffic to other proxies that continue to provide the service. The capacity of the cluster is sufficient to handle such failures.

Another example of isolation and failover is at the network level. In each PoP, the data center is connected to the Internet via different ISP connections. SD-NOC, a special software developed by Imperva, monitors the performance of each ISP connection and when it indicates that the connection failed or has serious performance issues, it will shut it down allowing the other ISP’s connections to take over.

Capacity Overprovisioning

This principle requires that we plan our system capacity in such a way that we have redundant resources even during the maximum expected load on the system. Through this principle, we can achieve required service capacity even during failure events.

This principle is always important when building highly available systems but in the case of Imperva Cloud services it is absolutely essential for defending against Denial of Service (DDoS) attacks as they, by definition, try to consume all resources in order to bring services down.

As such, we have designed our network and devices with capacity overprovisioning at the global and regional levels to meet the industry’s highest standards of three times the capacity of the largest DDoS attack volume observed. Combined with our advanced Behemoth technology, this allows us to provide an instant, seamless and zero-touch DDoS attack mitigation for our customers.

The same principle is applied for legitimate (“peace time”) traffic, where the entire PoP stack is designed and monitored with very strict requirements of low utilization to ensure our customers’ onboarded assets are provided with optimal performance.

Application reliability architecture principles

Interdependency

Dependency between applications is often essential, but we strive to minimize it as much as possible in order to increase the availability of the overall system. In other words, we build the applications in such a way that they will be as independent as possible from others.

For example, proxy applications, in addition to other real-time applications at the PoP, require configuration to know how to handle traffic. Routing, certificates, and custom rules are just a few examples of such configuration. Configuration files which contain this information are created by management applications which run in our centralized unified management console platform. We deliver these configuration files (typically within a few seconds) to each machine running a real-time application with the objective of making sure that those important applications have the necessary configurations on their local disk to provide the service without having to depend on the management platform or the connectivity to it. If we have to choose between having up-to-the-second configuration synchronization or the ability to be independent, we opt for independence.

Software / Feature Rollout

Software failures can cause as much or more damage than hardware issues due to their ability to impact many nodes at the same time. For example, a new software version could introduce a critical bug capable of impacting the service.

In order to decrease the impact of a software bug, we roll out the new version gradually across our production environment. This way, any potential critical issues will be detected at the very initial deployment stage. We deploy new revisions to our software periodically. The period could be up to two weeks depending on the criticality of the component. Generally, for critical components it takes two weeks to upgrade all software instances in production. We start with a small number of instances, accelerating as we gain confidence in the new release. New version rollouts are done automatically but stopped in the event of a problem.

When we introduce a new feature, we enable it in phases starting with our own websites which are protected by the service, followed by our beta customers and then finally all other customers. It is important for us to get feedback from a few customers before we introduce the feature to our entire customer base. The rollout of a feature can take a few weeks, depending on the risk we associate with the change. We control the rollout scope of a feature by configuration.

Configuration rollout / sandbox

Configuration, very much like a software, may impact the functionality and hence configuration changes introduce risks too. We separate configuration into two types using a different mitigation methodology for each type:

Global configuration: These are configuration settings that are applied to all of our customers' accounts. For example, out-of-the-box WAF security rules that help protect all of our customer websites. Such configuration updates are extensively tested before being applied to production (see the Software Quality Assurance section below). The change is rolled out like a software update, gradually through our entire network.

Customer-specific configuration: Our customers may tune their own configuration via GUI or via APIs. In both cases the

management application is designed to verify that the configuration entered (e.g., custom security rule) is valid and only then publishes the configuration to the PoPs.

Customer-specific configuration is written to different configuration files which only impacts the customer making the change and not the entire customer base. We deliver such changes to all PoPs within a few seconds, as the customer expects the changes to be applied without delay. As a result, we don't have time to do a slow graceful rollout as we do with global configuration.

We use additional methodologies to mitigate risks:

In the rare event where validation of the management application fails to catch an issue and invalid settings reach the PoP, the real-time component will have its own validation. If the settings are determined to be invalid, the real-time software will ignore these settings and trigger an error to our logs. These logs are reviewed by developers who investigate the issue, work to fix the bug in the validation, and work with the customer to amend their settings.

We leverage a sandbox methodology to avoid a scenario where all PoPs are impacted at once to create a massive outage. By publishing each configuration file to a sandbox "PoP" that is not being used by real customers before publishing to a real PoP, we can ensure that the software is not going to crash due to the new configuration. Only when successfully applied will the file be published to our entire network.

Self-healing software

We build our software in a distributed way incorporating several layers of protection against the threat of software bugs:

- A gradual software upgrade: reduces the impact of bugs in production
- A gradual global configuration rollout: reduces the risk of all customers being impacted in one configuration roll-out
- Sandboxing new customer configurations: ensures software does not crash as a result of a new configuration

The above layers of protection still may not cover 100% of cases and to address this we have developed self-healing software for critical services which works using the following mechanisms whenever software crashes and restarts.

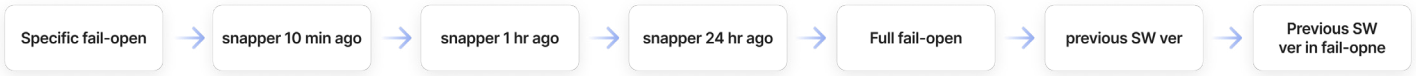
Snapper: In the event of a crash, the root cause could be a configuration update that triggered a bug. The service tries to use previous configuration versions, starting with the version from 10 minutes ago, then 1 hour ago, and finally from 24 hours ago. In parallel, errors will be sent to developers so they can investigate the issue and resolve it.

Fail-open mode: Sometimes the bug does not relate to a configuration at all and the only way to avoid the crash is to avoid running the code that is causing the crash. Since most of our code is related to security algorithms, we may choose to disable some of our security engine functionality in order to maintain sending traffic to the website. The result is a small reduction in security while still maintaining full operation and traffic flow. We first apply this change for the specific account where the crash was detected so no other customers will experience degradation in functionality. Only in the event that this mitigation fails will we disable the functionality for all accounts. This functionality applies to each proxy independently.

Using older software version: If nothing else fixes the issue, the application will try to use a previous software version in order to avoid any bugs introduced in the latest version.

The self-healing algorithms seldom need to be executed. However, it is important to have them in place in order to protect from very rare cases which may cause a large impact to the service.

The following diagram summarizes the different mechanisms used by the software for self-healing:



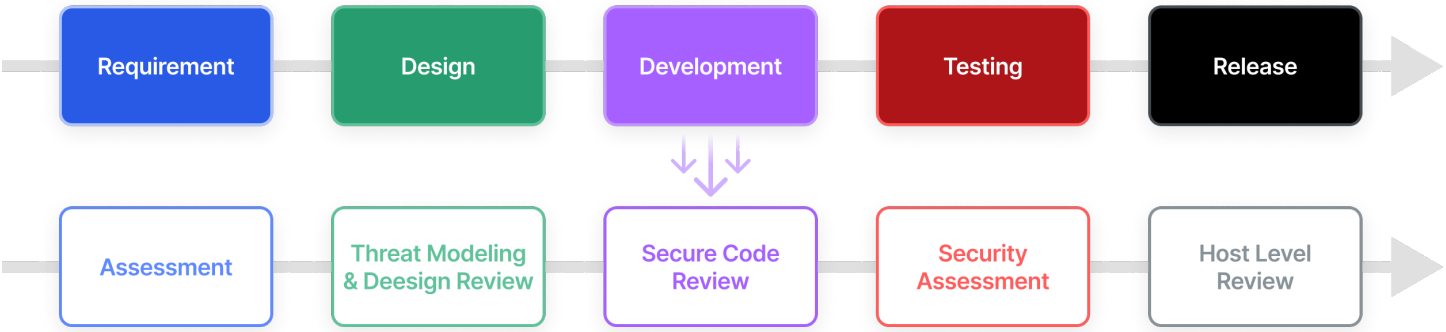
Software Quality Assurance

Coding practices

The Imperva Product Development team prides itself on continuously delivering high quality software. It starts with writing code following best practices in an organized, secure software development life cycle (SSDLC):

- A Product Requirements Document (PRD) is created and is carefully reviewed by Product Management, software architects, Product Development, Test Engineering team members, and representatives from the field to make sure the requirements will meet our customers' needs.
- Design is done by the development team and reviewed by the above forum and other relevant teams if needed.
- Any change in code is reviewed by a peer developer or a manager.
- Code review including security architecture review and penetration testing are performed throughout the process.

Security Processes in the SDLC



Fully automated regression

Imperva delivers new features constantly. Our release cadence depends on the software component, starting from bi-weekly deployment for more sensitive services ranging through weekly deployment and even agile deployment whenever needed.

In order to maintain high quality, we invest heavily in automatic regression testing that covers the functionality of the product. Those tests are performed almost daily so that when new code is added, it does not negatively impact existing functionality in any way. Success of those tests are gates on the way for new releases to production. Adding regression testing is part of any new feature.

Focus on unit & component testing

Our philosophy is that each development team should be entirely responsible for its services, which includes testing. The development team is responsible for writing its own component and unit testing.

End-to-end testing

Features and services sometimes contain multiple components. For high quality software it is essential not only to test each component by itself, but also to test the functionality end to end.

For this task we have a special staging environment where we run end-to-end regression testing nightly. We monitor errors in those tests and notify the appropriate team after the system debugs the error.

We have special Test Engineering teams who focus on writing, maintaining and running those tests and make sure bugs are fixed before release goes out.

Change Management Process

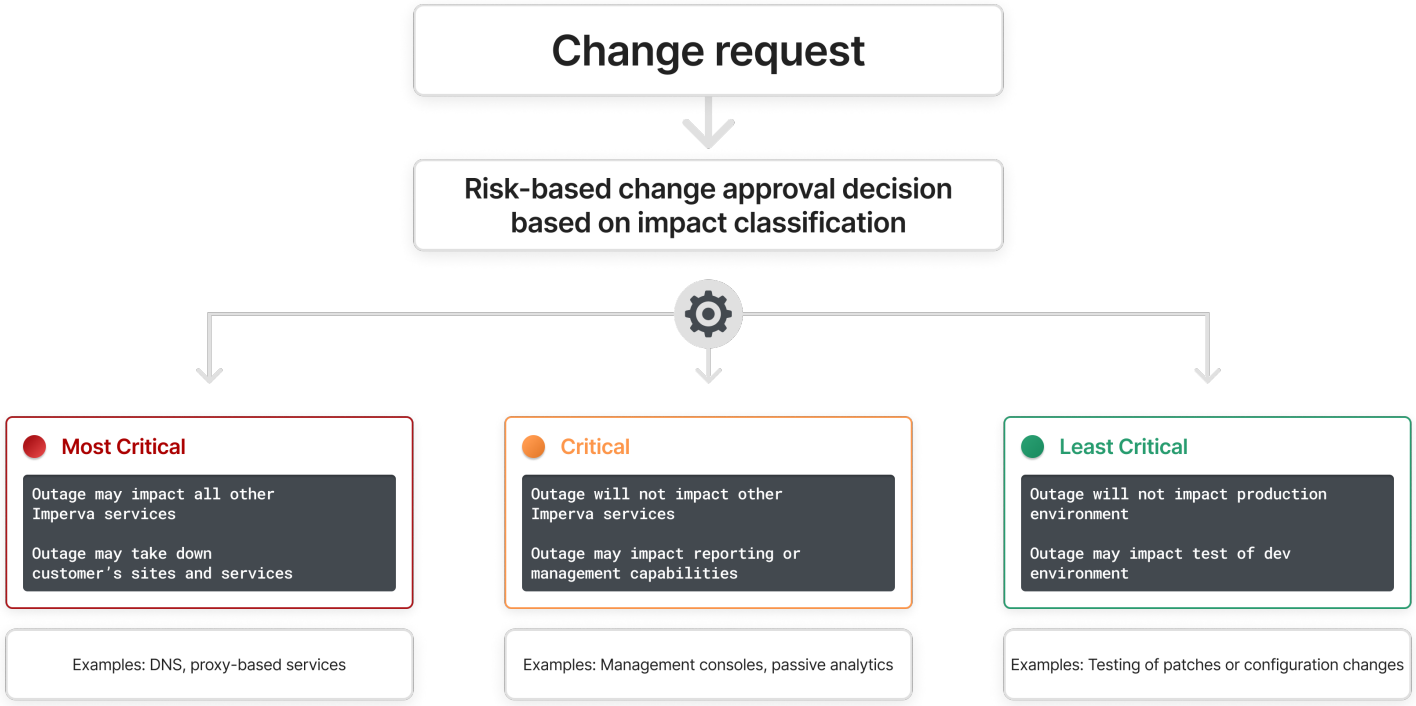
Impact classification and prioritization

The goal of the Change Management process is to implement planned and unplanned changes (both code and non-code) in a timely manner. In addition to aligning our change management program with leading cloud providers, Imperva uses a risk-based controlled methodology to apply proper checks and balances throughout the change management process.

When making changes to core code or systems, we consider the impact on the delivery of our cloud products. We also monitor third party providers and consider how their planned or unplanned outages could potentially impact services to customers.

Imperva's cloud solution is composed of 60 components or microservices. Based on service dependencies, outages may impact the customers' services or other Imperva services. Our change management process and monitoring prioritization take the impact classification into account.

We classify each of the SW components based on the level of risk incurred when introducing a change to the component. Each change request is evaluated based on the Reliability Service Level Objective (SLO) (99.999%, 99.95% etc.), test coverage and incident history. Depending on the criticality of the component, the degree of risk will determine the level of Change Management review needed to be carried out to approve the change.



Feature rollout

As Imperva Cloud is a shared environment, rollout of new functionality is done gradually aiming to reduce the risk and collect feedback. For each new feature a rollout plan is defined. Following announcement in the product “Release Notes”, the rollout will be performed on a subset of the accounts or part of the real-time components in each PoP. The rollout approval is subject to the change management policy and being tracked and communicated via the change control board.

Infrastructure changes rollout

Infrastructure and configuration changes follow the same change management process and approval as the application/feature.

These changes include changes to the following:

- Physical/environmental and infrastructure changes
- Network changes
- Application software changes
- Database changes
- Patch management changes
- Configuration changes

Changes and the risks associated with them are always owned by the component manager responsible for the change. With all changes, like application changes, a Jira ticket of the change is opened and documented.

This includes: reason for change, risk, risk for delay, rollout, rollback, mitigation and monitoring, possible effect of the change, effect on other components, and time to be executed. The rollout approval is subject to the change management policy and being tracked and communicated via the change control board.

Operations focused on reliability

Network Operations Center

The Imperva Global NOC (Network Operations Center) is a 24×7×365 monitoring and response team that is responsible for the overall health and operational success of our Cloud infrastructure. This function plays a key role in our commitment to customer success by continuously monitoring for DDoS attacks, ISP connectivity issues, and infrastructure health. NOC Engineers are staffed in a follow-the-sun rotation and are available to intervene with necessary actions that may be required to ensure the service remains operational with minimal impact to customers. Work within the NOC is performed according to well-defined playbooks and members of the team are thoroughly trained in our technology to make judgment calls as may be necessary.

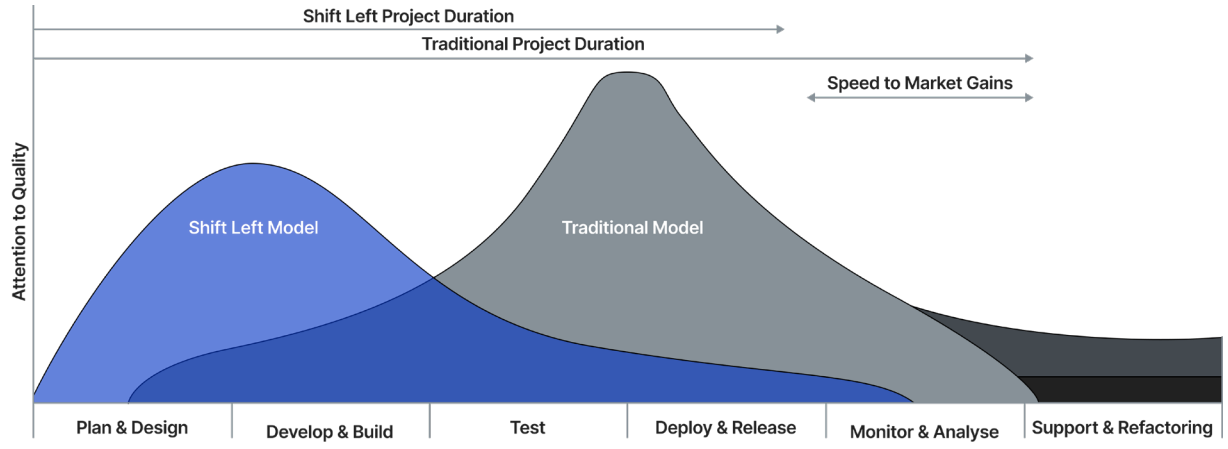
As explained in the “capacity overprovisioning” section, at times there may be a need to move traffic from one PoP to another in order to make sure they are not congested and impacting customer performance. The NOC performs such traffic shifts, or when needed, for example to allow maintenance activities performed at the PoPs.

Shift left - Developer-led quality and security testing

In Imperva, we have undertaken significant work in the pursuit of the “shift left” methodology of development-led code quality and security. The Imperva Cloud Web App and API Protection (CWAAP) platform comprises multiple services owned and developed by discrete, autonomous teams within the product development group.
Imperva places a lot of trust, and responsibility, in our development teams. Each team is expected to assume ownership of developing, securing, building, deploying, and releasing software as well as running regression testing on their product or service.

This allows the team a holistic view of the product life cycle and optimizes their ability to provide top quality service to our customers. Each team is measured on service availability in order to ensure a continuously high quality level of service to our customers.

The platform and infrastructure (TechOps) group provide Infrastructure as Code (IaC) Cloud and on-prem platforms, Security as a Service (SECaaS), including Static Application Security Testing (SAST), Software Composition Analysis (SCA) & Dynamic Application Security Testing (DAST) tools, education & support (In alignment with our InfoSec group), and act as consultants to all teams in order to ensure consistent standards of quality, security and reliability across the entire service.



Monitoring

As explained above each of our teams are responsible for the lifecycle of the service(s) they own. In order to ensure the platform runs smoothly, it is part of the definition of new products to have application level monitoring to verify functionality is working as expected.

Imperva uses industry best-of-breed monitoring solutions and also an Imperva developed, scenario-based monitoring platform to perform continuous monitoring, high frequency monitoring or batch processing of events (as required by each application, platform or service). These approaches provide Imperva with flexible options on how best to harvest Service Level Indicators (SLIs) and perform trend analysis to measure Service Level Objectives (SLOs) for our customer facing services.

In the event of a customer impacting incident fail, we leverage the power of PagerDuty to rapidly notify and engage all required stakeholders, and initiate the incident management process described in the subsequent section.

Application and system logs generated by Imperva services are consumed via Cloudwatch, Graylog, and Datadog for log-based alerting and to identify problems automatically.

Incident Management

Incident Management is a key focus area for Imperva. In the event of a customer impacting incident we initiate a standardized Incident Management process in which a PagerDuty incident, a telepresence War Room and Instant Messaging channel are generated. All participants required to resolve the incident join the War Room as part of a rapid response team. This team includes an Incident Commander, who manages the response and public notifications via StatusPage, Skilled Participants (Subject Matter Experts), who investigate the issue, communicate to the War Room and resolve the issue and customer communications liaisons (Customer Success), who communicate directly to customers with active support tickets. The incident management team works to establish immediate mitigation of the impact to the customers, followed by a more permanent resolution to the problem. Imperva sends incident updates to customers via the status.imperva.com website, which is an integral part of the proactive response to incidents and the transparent communication about the status of our cloud services.

Problem Management

Mitigation and resolution of incidents is not enough. Imperva strives to be a learning organization, and to achieve this we follow the Problem Management process which takes a holistic approach to the Incident. The primary element of the Problem Management process is the Root Cause Analysis (RCA) which begins as part of the incident remediation and continues with investigation by the Tier 2 or Tier 3 teams whose aim is to identify the true root cause. Additional considerations are given to contributing factors during the incident that are not necessarily part of the root cause but may have contributed to the impact. The incident response is also reviewed to identify areas of improvement which can improve Imperva's incident response and reduce the impact to customers during future incidents.

A postmortem meeting with the key internal stakeholders is also held within 3-4 days to agree on and implement preventative measures. These measures or 'Lessons Learned' are published across the development and operation organizations and highlighted in weekly cross-organizational meetings. The outcomes of the Problem Management process are communicated to the customer by the Imperva support team. Work begins immediately to implement short-term preventative measures while long-term preventative measures are scoped as projects and prioritized alongside new features in the bi-weekly and quarterly

planning processes. Each team reserves some of its time for implementing tasks that will improve reliability and prevent future incidents.

Summary

As digital transformation picks up pace, the opportunities for disruption by cyberattackers targeting web applications also increase. Organizations need to be certain they will have minimal disruption and continuity of business and services to their customers when faced with an attack or outages in the market.

Imperva's Cloud Application and Edge Security solutions are designed to stop these attacks and are built for reliability in such a way that customers can be assured of uninterrupted business operations in the event of an attack on their networks or applications.

This document outlines critical elements of development that make our solutions robust and dependable. It also delves into the processes applied during the development of Imperva's Cloud Application and Edge Security solutions to provide customers with the reassurance that they have chosen a service built for performance and reliability.

Among the areas covered are:

- The importance of a reliable network
- Platform reliability
- Application reliability
- Software quality assurance
- Change Management process
- Network operations
- Monitoring
- Incident Management and Root Cause Analysis

As the global cyber threat landscape evolves and application security grows as a priority for organizations, Imperva will continue to expand our network and enhance our services to protect our customers' critical infrastructure. For more information about any of our services visit www.imperva.com.

Imperva Service Level Objective (SLO) clarification

Imperva provides different Service Level Agreements (SLAs) to different architecture components based on their business impact. The table below documents the SLO, RPO (Recovery Point Objectives), and RTO (Recovery Time Objectives) of the dataplane and the management components.

Service Type	SLO: Downtime per year	DR Scenario/Recovery	RPO	RTO
Data Plane services ("Network Infrastructure") enables customers' websites to be protected and IP traffic to run	99.999% 5.26 minutes/year	Scenario: Complete PoP failure Recovery: traffic is automatically rerouted in the rest of Imperva's PoPs which is using anycast	0	0
Self-service Cloud console ("Peripheral Infrastructure") allows customers to view their website status and change their security settings	99.95% 4 hours 22 minutes /year	Scenario: AZ failure in AWS Recovery: Multi AZ redundancies per AWS best practices. The PoPs are unaffected and continue to pass traffic and enforce security policy based on their existing configuration	0	0